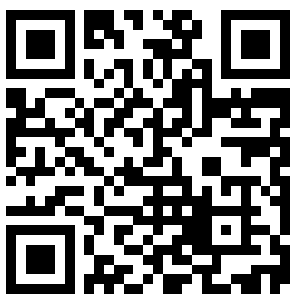


KR'92

Principles of Knowledge Representation and Reasoning

Proceedings of the Third International Conference

Cambridge, Massachusetts
October 25–29, 1992

Edited by:
Bernhard Nebel
Charles Rich
William Swartout

With support from the American Association for Artificial Intelligence, the European Coordinating Committee on Artificial Intelligence, and the Canadian Society for Computational Studies of Intelligence; in cooperation with International Joint Conferences on Artificial Intelligence, Inc.

Principles of Knowledge Representation and Reasoning:

Proceedings of the Third International Conference (KR '92)

The Morgan Kaufmann Series in Representation and Reasoning

Series editor, Ronald J. Brachman (AT&T Bell Laboratories)

Books

James Allen, James Hendler, and Austin Tate, editors
Readings in Planning (1990)

James F. Allen, Henry A. Kautz, Richard N. Pelavin, and Josh D. Tenenberg
Reasoning About Plans (1991)

Ronald J. Brachman and Hector Levesque, editors
Readings in Knowledge Representation (1985)

Ernest Davis
Representations of Commonsense Knowledge (1990)

Thomas L. Dean and Michael P. Wellman
Planning and Control (1991)

Matthew L. Ginsberg, editor
Readings in Nonmonotonic Reasoning (1987)

Judea Pearl
Probabalistic Reasoning in Intelligent Systems: Networks of Plausible Inference (1988)

Glenn Shafer and Judea Pearl, editors
Readings in Uncertain Reasoning (1990)

John Sowa, editor
Principles of Semantic Networks: Explorations in the Representation of Knowledge (1991)

Daniel S. Weld and Johan de Kleer, editors
Readings in Qualitative Reasoning about Physical Systems (1990)

David E. Wilkins
Practical Planning: Extending the Classical AI Paradigm (1988)

Proceedings

Principles of Knowledge Representation & Reasoning: Proceedings of the First International Conference (KR '89)
Edited by Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter

Proceedings of the Second International Conference (KR '91)
Edited by James Allen, Richard Fikes, and Eric Sandewall

Proceedings of the Third International Conference (KR '92)
Edited by Bernhard Nebel, Charles Rich, and William Swartout

The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Conference
Edited by Frank M. Brown (1987)

Reasoning About Actions and Plans: Proceedings of the 1986 Workshop
Edited by Michael P. Georgeff and Amy L. Lansky (1987)

Theoretical Aspects of Reasoning and Knowledge: Proceedings of the First Conference (TARK 1986)
Edited by Joseph P. Halpern

Proceedings of the Second Conference (TARK 1988)
Edited by Moshe Y. Vardi

Proceedings of the Third Conference (TARK 1990)
Edited by Rohit Parikh

Proceedings of the Fourth Conference (TARK 1992)
Edited by Yoram Moses

Principles of Knowledge Representation and Reasoning:

Proceedings of the Third International Conference (KR '92)

Edited by

Bernhard Nebel
(German Research Center for Artificial Intelligence)

Charles Rich
(Mitsubishi Electric Research Laboratories)

William Swartout
(University of Southern California/ Information Sciences Institute)

Morgan Kaufmann Publishers
San Mateo, California

Sponsoring Editor: *Michael B. Morgan*
Production Manager: *Yonie Overton*
Production Editor: *Carol Leyba*
Editorial Coordinator: *Douglas Sery*
Cover Mechanical: *Patty King*
Pasteup/Additional Composition: *Maryland Composition Company*

Most of the individual papers in this volume were author-typeset using specially designed macros for LaTeX.

Morgan Kaufmann Publishers, Inc.
Editorial Office:
2929 Campus Drive, Suite 260
San Mateo, CA 94403

95 94 93 92 4 3 2 1

Library of Congress Cataloging-in-Publication Data is available for this book.
ISBN: 1-55860-262-3

Contents

Acknowledgements

KR '92 would not have been possible without the efforts of a great number of dedicated people.

First, and most important, is our outstanding international program committee, who were asked to contribute extraordinary effort in reviewing and comparing over 300 papers, and who did an excellent (and all-electronic!) job:

James Allen
University of Rochester

Giuseppe Attardi
University of Pisa

Daniel Bobrow
Xerox PARC

Ron Brachman
AT&T Bell Laboratories

Gerd Brewka
GMD, Bonn

Rina Dechter
UC Irvine

Johan de Kleer
Xerox PARC

Jon Doyle
MIT

David Etherington
AT&T Bell Laboratories

Richard Fikes
Stanford University

Alan Frisch
University of Illinois

D. M. Gabbay
Imperial College

Michael Georgeff
Australian AI Institute

Patrick Hayes
Stanford University

Haym Hirsh
Rutgers University

Lewis Johnson
USC/Information Sciences Institute

Henry Kautz
AT&T Bell Laboratories

Kurt Konolige
SRI International

Craig Knoblock
USC/Information Sciences Institute

Gerhard Lakemeyer
University of Bonn

Maurizio Lenzerini
University of Roma

Hector Levesque
University of Toronto

Robert MacGregor
USC/Information Sciences Institute

Alan Mackworth
University of British Columbia

David Makinson
Paris

David McAllester
MIT

Fumio Mizoguchi
Science University of Tokyo

Wolfgang Nejdl
Technical University Vienna

Hans Jürgen Ohlbach
MPI Saarbrücken

Peter Patel-Schneider
AT&T Bell Laboratories

Ramesh Patil
USC/Information Sciences Institute

Judea Pearl
UC Los Angeles

Ed Pednault
AT&T Bell Laboratories

Martha Pollack
University of Pittsburgh

Jordan Pollack
Ohio State University

David Poole
University of British Columbia

Henri Prade
University Paul Sabatier

Erik Sandewall
University of Linköping

Karl Schlechta
IBM Germany

Len Schubert
University of Rochester

Stuart Shapiro
SUNY Buffalo

Lokendra Shastri
University of Pennsylvannia

Gert Smolka
German Research Center for AI

Lynn Andrea Stein
MIT

Devika Subramanian
Cornell University

Peter Szolovits
MIT

Mike Wellman
University of Michigan

Due to the large number of papers submitted to this year's conference, the program committee was fortunate to have the assistance of the following guest reviewers, whom they gratefully acknowledge:

Akira Aiba
ICOT, Japan

Jean-Paul Arcangeli
University Paul Sabatier

Franz Baader
German Research Center for AI

David Basin
MPI Saarbrücken

Alex Borgida
Rutgers University

Craig Boutilier
University of British Columbia

Hans Jürgen Bürckert
German Research Center for AI

Tom Bylander
Ohio State University

Phil Cohen
SRI International

James Crawford
AT&T Bell Laboratories

Gabriella Crocco
University Paul Sabatier

Jürgen Dix
University of Karlsruhe

Francesco M. Donini
University of Rome

Michael Freund
Université d'Orleans

Gerhard Friedrich
Technical University Vienna

Koichi Furukawa
ICOT, Japan

Moisés Goldszmidt
UC Los Angeles

Georg Gottlob
Technical University Vienna

Joe Halpern
IBM Almaden

Jochen Heinsohn
German Research Center for AI

Andreas Herzig
University Paul Sabatier

Anthony Hunter
Imperial College

Katsumi Inoue
ICOT, Japan

Walter Kasper
German Research Center for AI

Tadashi Kawamura
ICOT, Japan

Jörg-Uwe Kietz
GMD, Bonn

David Kinny
Australian AI Institute

Jerome Lang
University Paul Sabatier

Wiktor Marek
University of Kentucky

Jack Mostow
Carnegie Mellon University

Daniele Nardi
University of Rome

Andreas Nonnengart
MPI Saarbrücken

Steven Norton
Rutgers University

Werner Nutt
German Research Center for AI

Yoshihiko Ohta
ICOT, Japan

Massimo Poesio
University of Rochester

Anand S. Rao
Australian AI Institute

Stuart Russell
UC Berkeley

Klaus Schild
German Research Center for AI

Camilla Schwind
CNRS, Marseille

Bart Selman
AT&T Bell Laboratories

Liz Sonenberg
University of Melbourne

Gil Tidhar
Australian AI Institute

Christoph Weidenbach
MPI Saarbrücken

We would also like to thank our invited speakers for their important contributions to the conference program: Peter Szolovits and Martha Pollack, who will open the conference with their reports on how knowledge representation and reasoning research looks from the "trenches" of AI in medicine and natural language processing; Stephen Kosslyn of the Psychology Department at Harvard University, for his fascinating talk on "seeing with the mind's eye"; and Ray Reiter, for letting us schedule his not-to-be-missed talk on the Thursday afternoon to make sure people stayed

around until the end of the conference. We also thank Ramesh Patil for organizing the panel on knowledge sharing.

For local arrangements, we were very fortunate to have talked Jim Schmolze into a second year of service, assisted by the ever-competent Carolyn Miner of C. K. Miner & Associates. Between the two of them, we never had to worry about the logistical details of the meeting.

It is also our pleasure to continue our relationship with Mike Morgan, Doug Sery, and Carol Leyba of Morgan Kaufmann Publishers for the production of this proceedings volume.

As is traditional, the home organizations of the conference organizers provided most of the administrative support for this conference at no cost. In this case, we wish to thank the German Research Center for AI, and G. Jacquinot and D. Borchers personally; Mitsubishi Electric Research Laboratories, and Sandy Staff personally; and USC/Information Sciences Institute, and Kary Lau personally, for their invaluable administrative assistance. We would also like to give extra thanks to Mitsubishi Electric Research Laboratories for hiring the services of Wyn Snow to produce the conference brochure in LaTeX.

We gratefully acknowledge the generous support of the American Association for Artificial Intelligence, the European Coordinating Committee on Artificial Intelligence, and the Canadian Society for Computational Studies of Intelligence; and the cooperation of International Joint Conferences on Artificial Intelligence, Inc. We would like to personally thank Carol Hamilton, Annette Eldredge, Mike Hamilton, and Rick Skalsky at AAAI for their logistical help with mailing and publicity and coordination with the AAAI 1992 Fall Symposium Series.

Finally, we would like to thank Thea Nebel for waiting until a week *after* the reviews were done to be born.

Preface

The first Principles of Knowledge Representation and Reasoning conference, held in Toronto in 1989, was an experiment. With this, the third conference, we can safely say the experiment has been a success: "KR" has established itself as the leading forum for timely, in-depth (KR papers are twice as long as papers in the general AI conferences) presentation of new results in the theory and principles underlying the representation and manipulation of knowledge in computer systems.

KR' 92 continues the tradition of high standards and stringent reviewing established by the first two conferences. There was a record number of over 300 papers submitted to the conference this year, out of which the 67 in this volume were chosen for publication. The conference also continues to maintain its international character with 26 papers in this volume from authors outside North America.

Two important—and related—concerns of the conference committee this year were the need for more productive connections between theory and implementation and for more attention to representing knowledge in real domains. We believe the accepted papers this year successfully reflect a broadening of the conference in these directions.

The relationship between theory and implementation in our field is a complex one. Unlike application systems, whose ultimate purpose is (simply!) to satisfy some user community, research systems are implemented for a variety of purposes, such as to explore a new system architecture, experiment with a new inference procedure, or support a new kind of domain knowledge. It is very hard to write a good paper about an implemented system, one that brings out the underlying principles without getting lost in the details, yet grounds these principles in enough detail to be believable. The three-paper session on implemented systems at this year's conference (see papers by Brachman, Borgida, and Baader et al. on pp. 247–281) was a good beginning; we hope these papers will serve as role models for future authors.

There is a strong temptation in our field to stand above the nitty-gritty of specific domains on the high plateaus of pure logic. Unfortunately, this strategy runs the risk of irrelevance; we must pay attention to representing knowledge in real domains both to find out what the real problems are and to demonstrate that we have solved them. We are pleased that there are many papers in this year's conference that address knowledge representation issues in planning, diagnosis, or natural language processing and that focus on temporal, spatial, or numerical reasoning.

The largest single topic grouping this year (18 papers) is, as in previous years, nonmonotonic logic. For an insightful and witty commentary on this phenomenon, we draw the readers' attention to the abstract of Ray Reiter's invited talk, "Twelve Years of Nonmonotonic Reasoning: Where (and What) is the Beef?" at the end of this volume. The second abstract at the end of this volume summarizes the panel organized by Ramesh Patil on the DARPA Knowledge Sharing Effort. Knowledge sharing was a prominent topic at the 1991 conference; we predict it will continue to be an important topic at future conferences.

We hope this collection and the associated meeting will serve to advance the state of art in our field. We thank all of you who contributed papers, attended the conference, and read the proceedings.

Bernhard Nebel
Program Co-Chair

Charles Rich
Conference Chair

William Swartout
Program Co-Chair

I.

Planning and Temporal Reasoning

Declarative Knowledge Representation in Planning and Scheduling

Jacek Gibert*
Department of Computer Science
The University of Melbourne
Parkville 3054, Australia

Abstract

In this paper we present the declarative approach to applying knowledge representation systems to solving practical planning and scheduling problems. We propose a design methodology based on the theory of combining functional or logic programming languages within the framework of artificial intelligence. We describe a computer automated planning and scheduling system which has been developed for Qantas Airways Limited using this approach.

1 INTRODUCTION

Traditionally, two AI approaches are applied to problem solving in planning and scheduling. These may be referred to as classical planning and constraint based scheduling. The first approach aims to find the sequence of actions which transforms the planning problem, given by some initial state of the world, to a desired state which represents the solution. In the second approach, a solution is found with a given sequence of actions which can transform the initial state of the world into a constraint-defined goal state. For many practical AI systems, the application of classical planning is relatively simple and efficient, and many heuristic search algorithms can be readily adopted. In contrast, a lot of research effort is yet needed to find a method for solving constraint based scheduling tasks efficiently (Fox and Smith 1984), (Garlick and Orenstein 1990), and some theoretical issues of constraint based reasoning must be resolved (Guesgen *et al.* 1987), (Freuder 1989).

Some planning and scheduling problems can be directly formulated as a constraint satisfaction problem (CSP) (Keng and Yun 1989). CSP is a well researched field of AI (Shapiro 1987), in which planning and scheduling problems can be expressed precisely and completely as quantitative mathematical formulae. A factory scheduling problem, for instance, can be expressed in CSP. It might consist of a finite set of tasks which can be expressed as mathematical formulae. Each formula has a number of variables associated with it. Each variable has a finite domain of values representing the range of possible alternatives. A set of semantic constraints specifies the consistent choices of bindings for these variables. The goal is to find a valid simultaneous assignment of values to all variables in all constraints. The n-queens and other similar problems, where all constraints are known and fixed, are considered good examples of applications of CSP. However, this static approach to CSP is ineffective when applied to those practical planning and scheduling problems where one is faced with either an incomplete set, or unions of independent sets of changing constraints.

Recent research has departed from the static view of CSP to investigate dynamic CSP using advanced AI techniques of action routines or agents (Prosser 1989). Complex tasks such as scheduling, design or configuration, and any real-time planning, rely on solving dynamic constraint satisfaction problems. The key characteristic of dynamic CSP is that variables in constraints can be dynamically introduced or removed. Unfortunately, a major problem with dynamic CSP is that it lacks simple and clear declarative semantics, making it difficult for non-expert users to form dynamic constraints and apply them to achieve particular goals. The importance of clear semantics for CSP in dynamic domains has been demonstrated in (Georgeff and Lansky 1987) with the design of a planning system that combined both declarative semantics for constraint representation and operational semantics for constraint execution.

Constraint logic programming (CLP) languages (Jaffar and Lassez 1987), (van Hentenryck 1989), (Colmerauer 1990) naturally allow variables to be dynamically introduced into constraints. The CLP formalism provides a declarative framework for expressing dynamic CSP using general mathematical procedures such as the simplex algorithm (Jaffar and Michaylov 1987). The basis

* Present address: Division of Building, Construction and Engineering, CSIRO Australia, P.O. Box 56, Highett 3190, Australia (jacek@pirat.mel.dbce.csiro.au).

for the integration of constraint satisfaction and logic programming in CLP can be briefly described as follows:

(1) a resolution based inference engine is augmented with a single constraint satisfaction algorithm for testing satisfiability of constraints on a particular domain;

(2) a constraint satisfaction algorithm is applied to special types of predicates and the result of the application is obtained in the form of a substitution;

(3) the use of the inference engine is restricted to predicates that can be verified or falsified using only substitutions.

However, the practical constraint solving capabilities of CLP are often compromised by the trade-offs between the applicability, completeness and efficiency of the general mathematical procedure. Furthermore, the constraint solving procedure is deeply embeded in the language of logic programming. This raises the question of how to represent the expert's knowledge about the domain naturally and how to utilise multiple constraint solving procedures.

In our approach, which is oriented towards declarative knowledge representation, functional and logic formalisms are combined in a language called Functional Horn Clauses (FHC) (Gibert 1991). A complex dynamic CSP can be solved by expressing declaratively the procedural knowledge in the form of functional specifications, and the deductive knowledge in the form of logic specifications. The benefit of capturing the expert's knowledge about the domain this way is most evident in practical applications such as resource planning and scheduling (Tate 1985). The user also has capabilities of interactively defining special purpose constraints and controlling when and how such constraints are to be activated.

Based on this design methodology, we have successfully implemented the the air crew trip pairing operation (planning and scheduling of crew for a set of aircraft movements) for Qantas Airways Limited. Our declarative knowledge representation methodology is adaptable to meet specific user requirements, and the knowledge base can be maintained and modified by the users themselves. That is, users can manipulate knowledge as and when required on different levels of object abstraction by directly manipulating data objects and their attributes, by changing or declaring new functions on attributes, or by simply changing or adding attributed rules to a rule base.

Section 2 explains the interactive trip pairing process. Section 3 introduces attributed knowledge representation formalism. Finally Section 4 describes the implementation of the interactive trip pairing system.

2 INTERACTIVE TRIP PAIRING

The Interactive Trip Pairing (ITP) knowledge representation and reasoning system was developed to plan and schedule air crews for Qantas Airways Limited (Qantas). Qantas is Australia's largest international air carrier. It carries some 3.5 million passengers and covers 98 million kilometres annually. Qantas operates with 3,500 cabin crews (air stewards) and 1,000 pilots and other technical crew.

Matching air crews with aircraft flight times and routes is a complex planning operation. Industry and union regulations covering the working hours of these operators are extensive and very strict. It was also considered important to Qantas' industrial relations to allow the crew some input into scheduling. Previously the task had been performed manually by a group of planners, and the underlying philosophy of the ITP system was to enhance the skills of the planners with an interactive tool which would allow them to produce more efficient, cost effective and flexible scheduling patterns.

The interactive AI approach to planning and scheduling of crews and aircraft was seen by Qantas as a feasible practical alternative to an optimization approach based on linear algebra techniques, or an expert system approach based on procedural programming techniques. Although some optimization and expert systems can support basic knowledge representation and reasoning functions, they rely on the hard-coded algorithmic description of all solutions for a given problem domain. Optimization systems are limited to finding efficient solutions using preprogrammed local and/or global cost evaluation functions. Expert systems are limited to providing only advisory, decision and diagnostic support strictly implemented by the underlying imperative system. All too often, after the considerable effort of building an optimization or expert system for a practical planning or scheduling problem, new knowledge about the problem becomes available, rendering the system obsolete.

In its simplest form, trip pairing involves the allocation of air crew to aircraft. The planning operation can be viewed as a search for an acceptable assignment of available air crew to all scheduled aircraft movements, according to specified objectives, and in such a way that the pairing does not violate work rules negotiated by unions and regulations of aviation governing bodies.

Specifically, an interactive trip pairing process is defined as a set of actions or functions to be performed for a given aircraft schedule and a pool of available crews. Associated with each action is a set of semantic constraints which have to be satisfied before and after the action is performed. Constraints describe the set of resources that can be used to perform a given action, e.g. an air crew pattern is constrained by aircraft movement patterns.

Constraints also describe the time limits in which actions can be performed, e.g. within the scheduled departure and arrival times of aircraft. Deviations from the satisfactory air crew schedule are expressed as constraint violations. Constraints can be absolute (hard), e.g. all aircraft must have two pilots with ranks of captain and first officer, or preferences (soft), e.g. the third pilot may have the rank of second officer. The hard constraints usually represent legal crewing requirements, and violation results in a rescheduling of the affected air crew patterns. The soft constraints represent the crewing policies which could be used, for example, as heuristics.

We note that most trip pairing constraints are dynamic resource constraints because they are not only applied to the values of the resource variables of action predicates, but they are also rated on "relevance" to the solution. For example, the pilot with the rank of second officer may only partially provide a function of the third pilot on the 747-type aircraft and the flight engineer may be required to complete the legal crewing requirement. One can express this additional requirement by dynamically introducing a "new" variable and its constraints in the solution as a result of trip pairing the third pilot with the rank of second officer on the 747-type aircraft. Here the crucial problem is not just that the additional crew complement is required but that different complements for the same aircraft often need non-identical sets of additional crews. Similarly, some of the available crews often provide more than one function.

The initial design used selective information about the application domain. This prototype system was used for trials and training, and as more knowledge became available the system was easily extended and refined to become a full blown production system. The planning operation has been installed in three Qantas sites: Cabin Crew planning and scheduling, Technical Crew planning and scheduling, and Corporate Strategy planning, each having their own planning objectives and regulatory constraints.

3 ATTRIBUTED KNOWLEDGE REPRESENTATION

The formulation and handling of dynamic resource constraints is explained in this section. Even though our declarative formulation called attributed knowledge representation is theoretically not more expressive than other constraint reasoning methods, we gain practical expressiveness. Our approach makes it easy to understand constraint satisfaction, constraint manipulation, and coding up of a domain knowledge.

The implementation of interactive trip pairing uses hierarchical structuring of information along the lines suggested by Sacerdoti (Sacerdoti 1977). Plans are obtained as partial orders on planning and scheduling actions performed interactively over attributed object trees. The declarative semantics of the system is based on the declarative formalism of FHC (Gibert 1991). An equivalent operational semantics of the system is based on an incremental FHC computation of values of the attributes within an attributed object tree.

3.1 ATTRIBUTED OBJECT TREES

Attributes are associated with symbolic objects, which are hierarchically organized as *trees of attributed objects*. Attributes are used to specify information about objects within the tree. In particular,

- objects can be individual nodes or lists of other object nodes, eg. an object node *pattern* links a series of duty nodes (the time a crew spends on duty in an aircraft);
- attributes and objects have types which affect the way they can be attached to the tree, e.g. an object *pattern* has an attribute called *base* for which the type is *port* (crew home airport). Both the pattern and the port are objects, and the attribute links the two.
- some objects and attributes are defined externally to the tree. They provide an effective means of interaction with the outside world. For example *port* is an external object which has number of external attributes such as a *name* or a *code*., e.g. Tullamarine Airport as a *name*, and MEL as a *code*. These are defined in a separate ports database, rather than defined by the user in the tree.

Attributed tree structures are often found to be a computationally effective formalism for capturing the inherent dynamics of knowledge about the physical world. Attributed trees have been successfully used as data structures in syntax directed translations based on attributed grammars (Deransart *et al.* 1988). They have also been applied to automata and natural language processing, functional and logic programming.

Our motivation for using hierarchically organized attributed tree structures is the need for the ability to maintain conceptual order of knowledge representation. For instance, an experienced planner organises his or her trip pairing knowledge in such a way that crewing patterns covering certain ports are dependent on various other patterns covering these ports. In particular, a pattern can consist of a number of crew members - pilots and/or cabin crew - and remains constant for the duration of an aircraft's movements. The nature of aircraft crewing is that not all members of the crew will remain together for the duration of their trip. A pattern is established by the Qantas planners, and consists of the maximum number of crew members that will remain together until they return to Sydney on completion of the aircraft movement. The number and roles of the members of the pattern is the crew complement.

To illustrate, assume the total crew requirement of a particular aircraft is three pilots and five cabin crew. The flight originates in Sydney and ends in Los Angeles. On reaching Honolulu, two pilots and all the cabin crew continue on to Los Angeles, but the third joins another flight returning to Sydney and is replaced by a fourth pilot. One pattern will therefore consist of the two pilots and five cabin crew, the second of the pilot who returns to Sydney and the third pattern will consist of the fourth, replacement pilot. The total crew requirement can be satisfied by pairing or listing the appropriate patterns to the aircraft movement. However, the sum of all crew complements may not exceed the total requirement of that particular aircraft.

Two constraints need to be satisfied in this example:

- pattern names must be unique,
- the sum of the crew complements of all the patterns assigned to the operation must be less than or equal to the crew complement required for the operation.

These constraints impose dynamic semantical dependencies between pattern objects in an attributed tree of patterns. That is, only certain types of patterns can be appended to the tree, and specifically those patterns which have unique names and their complements are less than or equal to the complement required to crew the aircraft. Furthermore, the dependencies can change each time a new pattern is appended - a clear example is that the crew complement requires to be changed each time a new pattern is added to the tree.

More formally, the structure of the attributed tree of pattern objects imposes a partial order on a solution space. A particular attributed tree of pattern objects is a solution if the complement required to crew all aircraft are filled. Attributed trees provide declarative formalism for describing dynamic constraints on persistent objects such as patterns, which undergo modifications of their attribute values, as well as on tree structures. Therefore, a trip pairing problem based on attribute evaluation can be solved interactively, allowing the user to quickly find acceptable - rather than optimum - solutions.

When searching for a solution interactively, a sequence $P_1, P_2, \ldots$ of patterns is constructed, each P_i being a pattern object. A simplified trip pairing operation is defined, called *crew_aircraft*$_{\{i\}}$, which involves a single aircraft movement. This operation is indexed with a number of *pattern* objects, which are paired to the designated object called *aircraft movement*, in the form of a list. Each pattern object is denoted by an object variable *Pattern*. A pattern has an attribute *name* and a *crew complement* associated with it, describing the pattern name and the crew break down by role and number of crew members respectively. The *crew_aircraft*$_{\{i\}}$ operation has an attribute called *r_crew* which value can be expressed in terms of an attributed function

$$r\_crew : Pattern^{\,i} \rightarrow C$$

where $Pattern^{\,i}$ denotes the Cartesian product of all pattern objects on the list of *crew_aircraft*$_{\{i\}}$, and C is a set of all possible *crew complements* that could be assigned to this aircraft movement. This function calculates the crew requirement for the aircraft movements paired with this series of patterns.

In this example of the trip pairing operation we consider two semantic constraints for *crew_aircraft*$_{\{i\}}$: the name uniqueness constraint and the crew complement constraint. In Figure 3.1.1 the *crew_aircraft*$_{\{i\}}$ operation is represented by a rectangle connected by one edge to an attribute node and another edge to a null terminated list of pattern nodes. The pattern node is represented by a circle connected to the two attribute nodes and subsequent pattern node (denoted by the *null* symbol [] in this case). Attributes are represented by oval nodes which hold two pieces of information: either the name of the attribute or the pointer to the function that computes its value, and the last value computed. In our example, *name* and *p_crew* are attribute names, *p_crew* is the crew complement, and *r_crew* is a function pointer. The second semantic constraint is represented as a shaded line with a node representing the computation necessary to check the constraint.

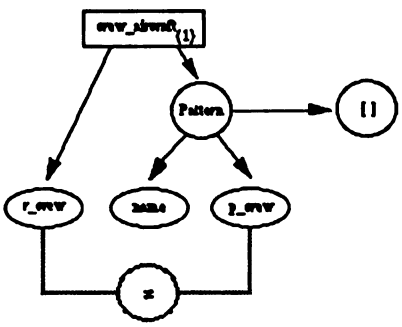


Figure 3.1.1 Object Dependency for a Single Pattern

Only the crew complement constraint is shown in the above figure, and because there are no other patterns, the name uniqueness constraint is satisfied.

Figure 3.1.2 demonstrates a sample attributed object tree of an operation after the addition of a second pattern to crew the aircraft. Both patterns now share the *r_crew* attribute, as the crew requirement for the aircraft movement might change after the addition of the new pattern. In order to check the crew complement constraint for the patterns, the values of *p_crew* attributes of the patterns are added and then compared with a possibly new value returned by *r_crew* function.

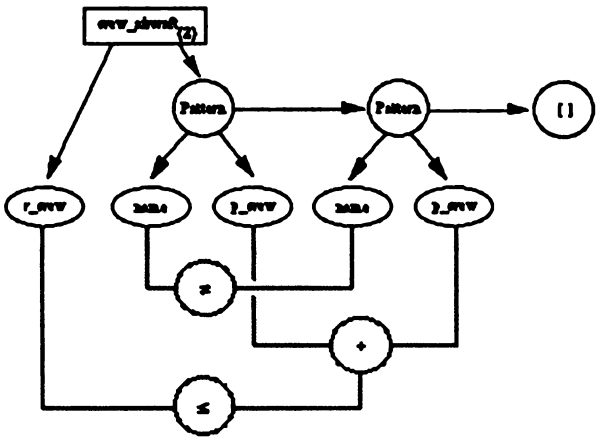


Figure 3.1.2 Object Dependency for Two Patterns

When another pattern has to be considered because some constraint was invalidated for the previous pattern, it is preferable to reuse as many as possible of the valid computations applied to the entire tree so far. However this means that we have to withdraw those computations done for last invalid pattern and all the conclusions drawn from them. This process can be quite complicated and the application of additional truth maintenance mechanisms is required (Bessiere, 1991). Using hierarchical structure of attributed trees guarantees the correct order of knowledge acquisition and significantly simplifies the process of truth maintenance.

3.2 DECLARATIVE REPRESENTATION FORMALISM

A declarative representation formalism can be a powerful tool for the systematic description, generation and maintenance of complex knowledge based systems. The definition of the attributed representation formalism is based on declarative formalism of a combined functional and logic programming language called Functional Horn Clauses (Gibert 1991). Our approach involves an application of the FHC computation to the evaluation of values of attributes over a global hierarchically organized tree structure of objects of the application domain. The strength of this approach comes from its ability to separate knowledge representation from inference strategy which allows for a declarative style of specification for both procedural and deductive knowledge.

Declarative knowledge specifications using combined functional and logic programming languages can sometimes be unusable as the computational formalism of a knowledge based system because they can result in excessive, even infinite computations. FHC solves this by allowing declarative knowledge specifications to be formulated with clauses containing recurrence expressions, which obey the global and hierarchical organization of attributed trees.

It is a feature of FHC that recurrences replace general functional expressions in logical clauses. This guarantees the existence of least Herbrand models and declarative semantics (Apt and van Emden 1982), (Gibert 1991). It can be shown that the process of attribution of tree-structured objects in FHC also has declarative semantics. Proof of this requires an extension of the declarative semantics of FHC to cover functional dependencies of attribute values in clauses.

The declarative representation formalism presented in this paper can be informally described as constraint based reasoning on representations of object properties by attributes. That is, attribute values are constrained by clauses and and computed by recurrences. Clauses represent relations between attributes of recurrence enumerated objects.

Formally, the structure of the attributed object tree is determined by the disjoint union of defined and external attributes of objects as follows. First, let *Obj* and *Attr* be pairwise disjoint sets of names of objects and attributes respectively. *Val* is the sum of values of all attributes of relevant types. *Dom(A)* denotes domain of an attribute set *A*. *Decl* denotes declaration mappings which associate a set of attribute names with each object name. Let ϕ be a recurrence mapping which assigns a set of values to a name from *Obj* and a non-negative integer, such that for any non-negative integer i, the set Obj_i contains all names of index i. Now for every set of attributes A of $Decl(Obj_i)$ there is a set of attributes A' of $Decl(Obj_{i+1})$, satisfying the condition $Dom(A) = Dom(A')$.

Next, let X be an object variable, then use $a\hat{}X$ to represent the attribute field a of some object X. We can extend FHC terms to *attributed terms* where:

- sub-terms can be labelled using attribute names,
- fixed arity is not required,
- the syntactic distinction between function names and attribute names can be removed, and
- annotated object variables can indicate reference to arbitrarily structured objects.

Due to the nested structure of the attributed terms, a hierarchy of objects can be declared. For example, in the case of the attributed term $b\hat{}(a_1\hat{}X_1, ..., a_n\hat{}X_n)$, attributes $a_1, ..., a_n$ are components of the attribute b. Compound attribute names can be used to name the components of an attributed term $b\hat{}X$. A component $a_i\hat{}X_i$ of an attribute term $b\hat{}X$ can be denoted as $a_i\hat{}b\hat{}X$ within the scope of the definition of the variable X as the structured object $(a_1\hat{}X_1, ..., a_n\hat{}X_n)$.

A *recurrence hierarchy* of attributed objects is defined as follows: a finite ground attributed term is a simple attributed object tree. An attributed object tree with index $n \geq 1$, is denoted by $\mathcal{T}_n$. Any ground attributed term belongs to a recurrence structured attributed object tree with index $n=1$. Assume that we can construct $\mathcal{T}_{i-1}$ from some attributed term t with the recurrence mapping ϕ. For every recurrence object X occuring in t we define the value of an attribute a as follows:

$$a\text{^}X\{i\} = \phi_i(X\{i\text{-}1\})$$

where the mapping ϕ_i specifies a hierarchical functional dependency of attribute values for X. Thus we construct $\mathcal{T}_i$. Furthermore with each recurrence object X we associate its cartesian closure X^* with the recurrence mapping ϕ. Such closure is only defined when each mapping ϕ_i, $0<i\leq n$, is also defined. But given each mapping ϕ_i, $0<i\leq n$, is total then the recurrence hierarchy can be uniquely defined by taking the cartesian closure of the domain of attributed object trees.

Let an *attributive representation formalism* be defined as a tuple

$$\mathcal{A} = [Obj, Attr, Val, Decl, Dom, AF, AR]$$

AF is the set of attributed functions or recurrences. Each element of AF is a multivalued function of multiple arguments from Val into Val. AR is a set of attributed rules or clauses which express constraints on values of attributes of objects.

Definition: A *specification* in attributive representation formalism $\mathcal{A}$ is a pair $(\mathcal{P}, \mathcal{R})$ consisting of a set of clauses $\mathcal{P}$ of AR and a set of recurrence relations $\mathcal{R}$ of AF for a hierarchy of attributed objects $\mathcal{T}_n$ built over $\mathcal{A}$ as follows: there is an atom p' in $\mathcal{P}$ and a ground substitution σ, such that

(1) if $n=1$, then $p = p'(X)\sigma$ for every object X in $\mathcal{T}_1$,

(2) if n>1, then $p = p'(X^*)\sigma$ for the cartesian closure X^* of every object X in $\mathcal{T}_n$ with the recurrence mapping ϕ,

and p is a logical consequence of $(\mathcal{P}, \mathcal{R})$.

The graphical example of the *crew_aircraft*$\{i\}$ operation can now be specified in our representation language as follows:

```
TYPE crew_aircraft* :: LIST;
TYPE pattern^crew_aircraft* :: PATT;
TYPE r_crew^crew_aircraft* :: (INT, INT, INT);
DEF pattern := (name^pattern, p_crew^pattern);
DEF pattern^crew_aircraft{i} := [ pattern |
      crew_aircraft{i-1}];
DEF crew_aircraft{0} := [] ;
DEF r_crew^crew_aircraft{i} := r_crew^crew_aircraft{i-1}
      - p_crew^pattern^crew_aircraft{i};
DEF r_crew^crew_aircraft{0} := (1,1,1) ;
DEF unique(name, pattern_list) :=
      FOR_ALL P::PATT IN pattern_list
            name^P <> name.
```

for type declarations, function and attribute definitions. The constraints are expressed declaratively by the following clause:

```
Pair(pattern^crew_aircraft*) :-
   r_crew^crew_aircraft* >= p_crew^pattern,
   unique(name^pattern, crew_aircraft*).
```

The following sequence of goals could be used to perform the *crew_aircraft*$\{i\}$ operation:

```
?- Pair( (ak01, (1,0,1))^crew_aircraft{1} );
yes
   ( r_crew^crew_aircraft{1} = (0,1,0),
     pattern^crew_aircraft{1} = [ (ak01, (1,0,1)) ] )

?- Pair( (bk01, (0,1,0))^crew_aircraft{2} );
yes
   ( r_crew^crew_aircraft{2} = (0,0,0),
     pattern^crew_aircraft{2} = [ (bk01, (0,1,0)) ] )
```

The declarative semantics of the above specification can be obtained by defining all functional dependencies and logical constraints which characterise possible values for the attributes of objects in the attributed object tree. For instance, the declarative meaning of functional dependencies of attribute values *r_crew^crew_aircraft*$\{i\}$ is expressed using the recurrence definition. The declarative meaning of *Pair(pattern^crew_aircraft*)* is expressed as a cartesian closure of a set of logical consequences of predicate formulae *Pair(pattern^crew_aircraft*$\{i\}$) ranging over all attribute values and object variables of the recurrence defined list *pattern^crew_aircraft*$\{i\}$. That is, the clause implications must always be true for all currently computed values of the attributes of the objects in the list.

3.3 OPERATIONAL SEMANTICS

The fundamental computational structure of operational semantics is that of a recurrence attributed object tree. In the tree, objects are connected by typed links. These links connect disjoint sets of attributes of a recurrence defined series of objects such that a set of attributes of the object in the series depends on attributes of the preceding objects of the series. We can use a bottom-up attribution algorithm to successively evaluate the attributes in each

series, starting with the first object of the series. Each attribution step takes place in the context of the hierarchical structure of other recurrence object series, and uses evaluated attributes from these series.

For example, in the crew pairing operation, *patterns* are composed of *duties*, which in turn are composed of *sectors* etc. Each object in the series within the hierarchy has its own set of attributes associated with it. All attributes within a series of objects can also be grouped into their own attribute series.

The soundness of the operational semantics is a direct consequence of the hierarchical structure of a recurrence attributed object tree. However the completeness can only be guaranteed in those cases where we can statically analyse functional dependencies between attributes of objects in the tree in order to determine the grouping of attributes into recurrence series. In general, as is the case with general functional computations, there can be fully recursive dependencies between objects in trees (e.g rational trees) which make it impossible to subdivide attributes into recurrence series. However, if functional dependency cycles between attributes are restricted to recurrences only, then we know from the declarative semantics that unique and consistent attribute values can be computed incrementally and systematically by evaluating series of attributes within a hierarchy of objects.

Flexible control of the bottom-up order of attribute evaluation is possible by way of structural change to the hierarchy of objects. For instance, the depth-first object hierarchy is efficient for computing values of linearly dependent attribute series, but if the recurrence dependencies are not linear then breadth-first object hierarchy is more desirable.

4 ITP SYSTEM DESCRIPTION

The operational role of the ITP system is to guide planners in drawing scheduling patterns on the screen in accordance with the rulebase. The system calculates duty hours and crew allowances and other information required in the planning process, which assists the planners to evaluate costs of patterns, effects of changes to the rule base, and the effects of changing arrival and departure times of aircraft.

There are two major parts to the ITP system: an attributed rulebase system and an application interface system. The attributed rulebase system is further divided into the attributed functions module and attributed rulebase module. The attributed function module uses functional programming language *AF* to define attributes, data objects and *AF* functions. The attributed rulebase module uses logic programming *AR* language to express rules. The application interface system consists of an interface module, system engine and interface functions. The interface module contains an application dependent front-end which drives the display, handles external data communication etc. The system engine drives the attributed data and rulebase and handles all data objects. The interface function module performs system level computations required by the attributed data and rulebase system.

Initial time to develop the prototype and undertake all field research was twelve man months. After nearly two years of implementation and refinement, the production system has been successfully installed at three sites in Qantas, and has been in operation for about two years. Adaptation of the system for another airline takes about six man months from user requirement specification to complete installation.

4.1 ATTRIBUTED RULEBASE IMPLEMENTATION

An important aspect in building the attributed rulebase system was that the expert knowledge of the planners did not have to be hard coded into the ITP system, but could be declaratively represented using an attributed knowledge representation language. As we have seen above, this representation language is able to express the dynamic structure of information about the planning and scheduling processes, and support user requirements to infer and reason about this information in a sound and complete manner. The *AF* and *AR* languages accept structured English statements, and the rules and functions can be easily altered by the users to explore "what-if" evaluations.

For example, the knowledge base is built using statements like

```
Sydney is a base for Qantas.
Transit through the base port should be avoided.
If pattern contains 2 short rest periods between
last 3 consecutive duties then
    the long rest period is required for this pattern
    after this duty.
```

These statements are interpreted as functions in *AF* (Attributed Functions) and clauses in *AR* (Attributed Rules) which act as both constructors of and constraints within the hierarchy of attributed object structures. For instance, the last statement from the example above determines and constrains the value of the attribute *rest period* of object *duty* in the series duty structure of object *pattern*. Note that the statement determines neither the value of *rest period* of the current object *duty* nor the value of *rest period* of the next object *duty* in the series when the next object *duty* is appended to the series of duty objects of the current *pattern*. Instead it simply characterises the current *pattern* as one needing a long rest

period before the next duty is appended. The value of the attribute *rest period* influences the construction of the duty object that can be appended to this pattern.

Declarative constructs used to build the rulebase closely resemble logic clauses:

Given that expression [and|or] expression then
    expression must be expression
otherwise expression.

This particular clausal form of the attributed rules has been chosen because of the efficiency of its sequential evaluation. The *must be* part of the rule usually represents an actual constraint, and is therefore evaluated first. If this part succeeds, the evaluation of the entire rule succeeds. Only if the *must be* part of the rule fails is the optional *Given that ... then* part of the rule evaluated. If the *Given that ... then* part of the rule succeeds, the rule is so far invalidated and therefore the optional *otherwise* part is evaluated. Evaluation of *otherwise* might succeed which will validate the rule, however if it does not then evaluation of the rule fails. If the *Given that ... then* part of the rule fails, then the rule is not applicable.

Rules are sequentially applied to a series of objects in the hierarchy. Rules can refer to a number of object series in the hierarchy, but can only be effectively applied to attributes of the current object in each series. Therefore all attributes of objects preceding the current object must already have been assigned values. If the rule is of the correct form, and the objects and their attributes mentioned in the rule exist, then the conclusion of the evaluation of the rules can be expressed as true/false. If the conclusion is true, the hierarchy of objects constructed to that point is valid for that rule. If the conclusion is false, two types of false conclusions can be drawn depending on the type of rule - a complete rule failure for a formal rule or hard constraint, or a warning for an informal rule or soft constraint.

Rule failures form a necessary part of any practical scheduling, and play a very important part in interactive plan generation for trip pairing. Inconsistencies of violated hard constraints can be corrected either by the user or by the system, and the differences in evaluation of hard and soft constraints can be similarly controlled. Therefore attributed functions can be related directly to rules in order to decide what to do in the event that the rule is found to be false. This can be done by using the *otherwise* connective in a rule. For example, we could have the rule as follows:

Given that pattern contains 5 rest periods
between duties then
    the last one should be greater than 2 days
otherwise
    recommend 2 day rest period for the last duty.

The complex meaning of *this rule* can be expressed declaratively as follows:

(1) If it is false that the value of the attribute *rest period* for the last duty object is greater than 2 days, and

(2) it is true that the given pattern object contains 5 rest periods between duties, then the rule so far is false.

Thus the value of the *otherwise* part of the rule must now be looked at to determine the logical value of the whole rule.

(3) If the *recommend 2 days rest period for the last duty* function returns true, then the whole rule becomes true.

However if this function returns false, then the whole rule is false.

The programming the attributed rulebase has parallels to object oriented programming. The user can simply declare the objects for consideration by the system, providing as much information as desired. When prompted, the system will then automatically compute all remaining values of the attributes of the objects in order to combine objects into an hierarchical structure.

The rulebase computation is implemented as an incremental process of walking through the hierarchy of the attributed object trees, then computing values for the object nodes of the trees and their attributes. The values are stored as indexed states within each node. A value in the state for a node becomes final only when an interpretation step has been completed successfully for the hierarchy up to this indexed stage. The indexed state of each node allows for an efficient "what-if" interpretation of the rulebase. Rules and functions are simply computed each time within the new instance of object nodes in the hierarchy. If the computation fails, that instance is returned to its previous state.

The correctness of the incremental interpretation process relies on successful completion of previous interpretation steps for all objects and attributes of the structure. However, if for some reason an object is to be removed from the hierarchy, the validity of the structure cannot be guaranteed. There are two ways to avoid this problem. The first is to reconstruct parts of the affected hierarchy and recompute the entire structure from the beginning. This method is computationally expensive. The second method is to provide a set of attributed rules and attributed functions to control and manage the removal of the objects from within the structure. Such removal rules have been successfully developed for the scheduling process for Qantas.

4.3 APPLICATION INTERFACE IMPLEMENTATION

The application interface system constitutes a large part of the ITP system. It consists of an interface module, system engine and interface functions.

The interface module contains an application dependent front-end, which is tailored to the particular application domain. It cannot be readily changed by the planner because changes must be made through the low-level external function call interface of the C programming language. However, it can be easily amended by the programmer with minimal knowledge of the workings of the ITP systems.

The system engine drives the attributed data and rulebase and handles all data objects. It provides a suitable development support environment, which helps the end-users formulate planning or scheduling operations within the data and rulebase system. The interface here consists of the rulebase interpreter with the interactive stepper. The stepper enables the user to monitor the evaluation of rules, attributes and functions, as well as view the attribution of object structures.

The rulebase interpreter was implemented using a standard combination of pattern/template matching, recursive grammar parsing and semantic attribute dependency analysis. The production ITP system integrates the data and rulebase system modules by means of the C compiler. The compilation of the system can be done in two stages: the data and rulebase compilation and the application interface compilation. First, the attributed functions and attributed rules from the data and rulebase module are compiled into the C programming language code using the purpose built compiler. Second, the resulting C code, the interface functions and the general system code is compiled into appropriate machine code using the C compiler provided by the vendor of the hardware. After a rulebase has been compiled into the system, user access to the ITP system is mainly through the graphical interface.

The interface function module performs system level computations required by the attributed data and rulebase system. Implementation of the interface system was designed to be as flexible as possible to allow development of different applications by building an appropriate data and rulebase system. Most interactive interface aspects of the system, from the graphical interface to the rulebase, can be modified or enhanced by way of the interface function module. The interface functions module provides an interface to external calls to functions in the C programming language and other low-level system functions, through which the data and rulebase system communicates. In addition, the interface functions module provides the capability to alter the physical implementation of data objects. This module allows for full integration of the production system implementation plus efficient tailoring of the system to the specific requirements of the hardware.

The appendix demonstrates some complex scheduling rules and illustrates the application interface specifically designed for Qantas. The screen shows a partially completed crewing pattern and pattern display window. Pattern information, including the times and routes of aircraft movements and crew details, is entered in the top right hand corner window. The pattern information is then displayed numerically in the window immediately below this, and is represented graphically in the spider graph in the top left hand window. Results of the evaluation of the rulebase are displayed in the lower right hand window. The control panel, on the lower left hand side, controls the parameters of the system, such as selection of objects, entry screens for external attributes etc.

User acceptance of the ITP system was dependent on an interface that was as easy to use as the current manual system, requiring minimal training or computer literacy, and a computational ability to complete the task in significantly less time. The system utilises screen buttons for navigational and functional purposes, and places graphical and textual displays at any screen location on a dual screen display, and intercepts user actions as mouse clicks on any screen.

5 CONCLUSION

An extensive range of implementations exist for planning and scheduling. These implementations are mainly based on optimization and procedural software-building technology. AI approach to planning and scheduling is becoming an acceptable practical alternative to the more traditional optimization approach. Some example implementations include the factory floor scheduling system ISIS (Fox & Smith 1984) or BHP's steel production scheduling system (Garlick and Orenstein 1990). The implementation of an AI based planning and scheduling system for Qantas Airways Limited follows a new knowledge representation approach to solving a resource planning and scheduling problems.

A declarative methodology for applying a knowledge representation system to planning and scheduling using the attributive knowledge formalism has been developed, where knowledge about an object in the application domain could be represented in terms of its attributes. This methodology is based on a combination of functional and logic paradigms and a notion of object attribution to describe complex outside world knowledge domains declaratively.

Not all aspects of the declarative methodology have been fully examined, and in particular whether it can be

effectively applied to other practical application areas dominated by special-purpose procedural methods such as real number constraint solving or linear algebra based optimization. This is a subject for future study.

Acknowledgements

Special thanks are due to numerous people at The Preston Group Pty. Ltd. and Qantas Airways Limited. They provided the opportunity for a commercial application of this research, and in doing so, have tested the research to the fullest. I am especially grateful to all the members of the development team who have helped to make the project a success.

References

Apt, K. and van Emden, M.H. Contributions to the theory of logic programming. *J. ACM 29*, 3 (July 1982), 841-862.

Bessiere, C. Arc-consistency in dynamic constraint satisfaction problems, In Proc *Tenth National Conference on Artificial Intelligence*, Anaheim California, 1991, pp 221-226

Colmerauer, A. Prolog III Universe. *Communications of ACM 33*, 6 (June 1990).

Deransart, P, Jourdan, M. and Lorho, B. Attribute Grammars. Definitions, Systems and Bibliography. *Lecture Notes in Computer Science 323*, Springer-Verlag, 1988.

Fox, M.S. and Smith, S.F. ISIS - a knowledge-based system for factory scheduling. *Expert Systems 1*, 1 (1984).

Freuder E.C. Partial constraint satisfaction. In Proc. *The Eleventh International Joint Conference on Artificial Intelligence*, IJCAI-89, Vol. 2, Detroit Michigan USA, August 1989, pp. 278-288.

Garlick, S. and Orenstein, B. The design and development of an intelligent scheduling assistant. In Proc. *Australian Artificial Intelligence Conference*, November 1990, pp. 517-535.

Georgeff, M.P. and Lansky, A.L. Procedural knowledge. Tech. Rept., Note 411, SRI International, Menlo Park California, January, 1987.

Gibert, J. *Declarative Symbolic Computations*, PhD thesis, Tech. Rept. 91/31, Melbourne University, Parkville Australia, 1991.

Guesgen, H.W., Junker, U and Voss, A. Constraints in a hybrid knowledge representation system. In Proc. *The Tenth International Joint Conference on Artificial Intelligence*, IJCAI-87, Milan Italy, 1987, pp 30-33.

van Hentenryck, P. *Constraint Satisfaction in Logic Programming*, MIT Press Cambridge Massachusetts, 1989

Jaffar, J. and Lassez, J.-L. Constraint logic programming. In Proc. *Conf. on Principles of Programming Languages*, ACM, 1987, pp. 111-119.

Jaffar, J. and Michaylov S. Methodology and Implementation of a CLP System. In Proc. *4th International Conference on Logic Programming*, Melbourne, May 1987.

Keng, N. and Yun, D.Y.Y. A planning/scheduling methodology for the constrained resource problem. In Proc. *The Eleventh International Joint Conference on Artificial Intelligence*, IJCAI-89, Vol. 2, Detroit Michigan USA, August 1989, pp. 998-1003.

Prosser, P. A reactive scheduling agent. In Proc. *11th International Joint Conference on Artificial Intelligence*, August 1989, pp. 1004-1013.

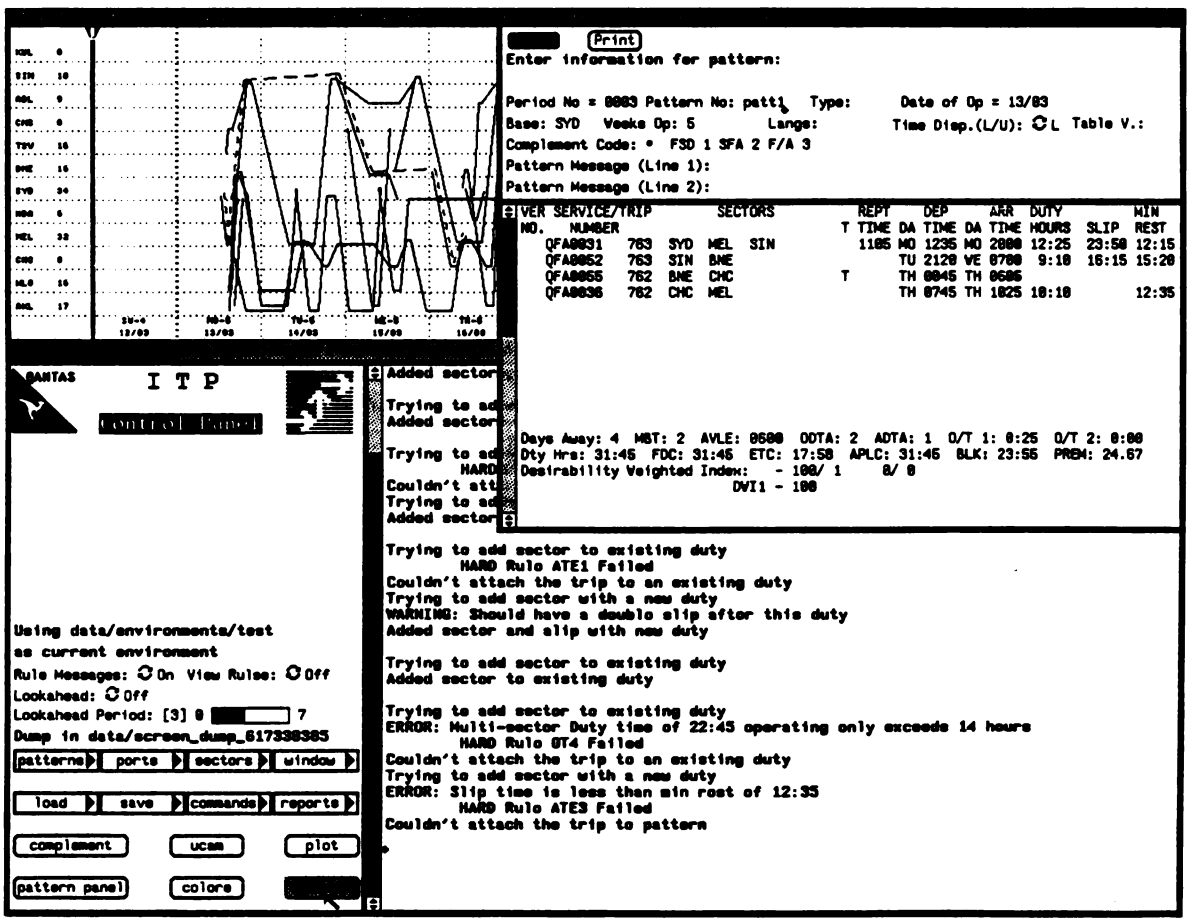
Sacerdoti, E. *A Structure of Plans and Behaviours*, North-Holland Amsterdam, 1977

Shapiro S.C. (editor) *Encyclopedia of Artificial Intelligence*, John Wiley & Sons, Chichester England, 1987

Tate, A. A review of knowledge-based planning techniques. In *Expert Systems '85*. Cambridge University Press, 1985.

Appendix

```
/* QANTAS Attributed Rules */

/* Ref. Work Rule A.1.(a).(i). */
OT4:Given that the Duty is Operating and the Duty is not Paxing and
      the Number_of_Sectors in the Duty is greater than 1
    then Duty_Time in the Duty must be less than or equal to 14 hours
    otherwise
      error "Multi-sector Duty time of %t operating only exceeds 14 hours",
            Duty_Time in the Duty .
/*
 * Decide if either to attach a trip to the current duty or to create
 * a new duty for the pattern.
 */
ATE1:Given that Arrival_Time of the Sector minus Departure_Time of the
       First_Sector in Duty is greater than 24 hours minus Sign_On_Time
       for the First_Sector in Duty minus Sign_Off_Time for the Sector
     then the Sector must not be Attached_to_End of Duty.

ATE3:Given that Departure_Time of the Sector minus Arrival_Time of the
       Last_Sector in the Duty minus Sign_On_Time for the Sector minus
       Sign_Off_Time for Last_Sector in Duty is less than Min_Rest for Duty
    then Duty must not be Attached_to_End of Pattern
    otherwise
        error "Slip time is less than min rest of %t", Min_Rest for Duty.
```



Intelligent Backtracking Techniques for Job Shop Scheduling

Yalin Xiong **Norman Sadeh** **Katia Sycara**

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Abstract

This paper studies a version of the job shop scheduling problem in which some operations have to be scheduled within non-relaxable time windows (i.e. earliest/latest possible start time windows). This problem is a well-known NP-complete Constraint Satisfaction Problem (CSP). A popular method for solving these types of problems consists in using depth-first backtrack search. Our earlier work focused on developing efficient consistency enforcing techniques and efficient variable/value ordering heuristics to improve the efficiency of this procedure. In this paper, we combine these techniques with new look-back schemes that help the search procedure recover from so-called deadend search states (i.e. partial solutions that cannot be completed without violating some constraints). More specifically, we successively describe three intelligent backtracking schemes: *Dynamic Consistency Enforcement* dynamically enforces higher levels of consistency in selected critical subproblems, *Learning From Failure* dynamically modifies the order in which variables are instantiated based on earlier conflicts, and *Heuristic Backjumping* gives up searching areas of the search space that appear too difficult. These schemes are shown to (1) further reduce the average complexity of the search procedure, (2) enable our system to efficiently solve problems that could not be solved otherwise due to excessive computational cost, and (3) be more effective at solving job shop scheduling problems than other look-back schemes advocated in the literature.

1 Introduction

This paper is concerned with the design of recovery schemes for incremental scheduling approaches that sometimes require undoing earlier scheduling decisions in order to complete the construction of a feasible schedule.

Job shop scheduling deals with the allocation of resources over time to perform a collection of tasks. The job shop scheduling model studied in this paper further allows for operations that have to be scheduled within non-relaxable time windows (i.e. earliest possible start time/latest possible finish time windows). This problem is a well-known NP-complete Constraint Satisfaction Problem (CSP) [11]. Examples of such problems include factory scheduling problems, in which some operations have to be performed within one or several shifts, spacecraft mission scheduling problems, in which time windows are determined by astronomical events over which we have no control, factory rescheduling problems, in which a small set of operations need to be rescheduled without revising the schedule of other operations, etc.

A generic approach to solving CSPs relies on depth-first backtrack search [24, 13, 2]. Using this paradigm, scheduling problems are solved through the iterative selection of a variable (i.e. an operation) and the tentative assignment of a value (i.e. a reservation) to that variable. If in the process of constructing a solution, a partial solution is reached that cannot be completed without violating some of the problem constraints, one or several earlier assignments have to be undone. This process of undoing earlier assignments is referred to as *backtracking*. It deteriorates the efficiency of the search procedure and increases the time required to come up with a solution. While the worst-case complexity of backtrack search is exponential, several techniques have been proposed in the literature to reduce its average-case complexity [7]:

- *Consistency Enforcing Schemes*:[15] prune the search space from alternatives that cannot par-

ticipate in a global solution . There is generally a tradeoff between the amount of consistency enforced in each search state[1] and the savings achieved in search time.

- *Look-ahead Schemes*: variable/value ordering heuristics[2, 14, 16, 7, 9, 20] help judiciously decide which variable to instantiate next and which value to assign to that variable. By first instantiating difficult variables, the system increases its chances of completing the current partial solution without backtracking [14, 9, 20]. Good value ordering heuristics reduce backtracking by selecting values that are expected to participate in a large number of solutions [7, 20].
- *Look-back Schemes*:[22, 8, 12, 5] While it is possible to design consistency enforcing schemes and look-ahead schemes that are, on the average, very good at efficiently reducing backtracking, it is generally impossible to efficiently guarantee backtrack-free search. Look-back schemes are designed to help the system recover from deadend states and, if possible, learn from past mistakes .

Our earlier work focused on developing efficient consistency enforcing techniques and efficient look-ahead techniques for job shop scheduling CSPs [17, 18, 9, 23, 21, 20, 19]. In this paper, we combine these techniques with new look-back schemes. These schemes are shown to further reduce the average complexity of the search procedure. They also enable our system to efficiently solve problems that could not be efficiently solved otherwise. Finally, experimental results indicate that the schemes described in this paper are more effective at solving job shop scheduling problems than other look-back schemes advocated in the literature.

The simplest deadend recovery strategy consists in going back to the most recently instantiated variable with at least one alternative value left, and assigning a different value to that variable. This strategy is known as *chronological backtracking*. Often the source of the current deadend is not the most recent assignment but one performed earlier. By changing assignments that are irrelevant to the current conflict, chronological backtracking often returns to similar deadend states. When this happens, search is said to be *thrashing*. Thrashing can be reduced using backjumping schemes that attempt to backtrack all the way to one of the variables at the source of the conflict [12]. Search efficiency can be further improved by learning from past mistakes. For instance, a system can record earlier conflicts in the form of new constraints that will prevent it from repeating earlier mistakes [22, 8]. Dependency-directed backtracking is a technique incorporating both backjumping and constraint recording [22]. Although dependency-directed backtracking can greatly reduce the number of search states that need to be explored, this scheme is often impractical due to its exponential worst-case complexity (both in time and space). For this reason, simpler techniques have been developed that approximate dependency-directed backtracking. *Graph-based backjumping* reduces the amount of book-keeping required by fullblown backjumping by assuming that any two variables directly connected by a constraint may have been assigned conflicting values [5]. *N-th order deep and shallow learning* only record conflicts involving N or fewer variables. [4].

Graph-based backjumping works best on CSPs with sparse constraint graphs [5]. Instead, job shop scheduling problems have highly interconnected constraint graphs. Furthermore graph-based backjumping does not increase search efficiency when used in combination with forward checking [14] mechanisms or stronger consistency enforcing mechanisms such as those entailed by job shop scheduling problems [20]. Experiments reported at the end of this paper also suggest that N-th order deep and shallow learning techniques often fail to improve search efficiency when applied to job shop scheduling problems. This is because these techniques use constraint size as the only criterion to decide whether or not to record earlier failures. When they limit themselves to small-size conflicts, they fail to record some important constraints. When they do not, their complexities become prohibitive.

Instead this paper presents three look-back schemes which have yielded very good results on job shop scheduling problems:

1. *Dynamic Consistency Enforcement* (DCE): a selective dependency-directed scheme that dynamically focuses its effort on critical resource subproblems,
2. *Learning From Failure* (LFF): an adaptive scheme that suggests new variable orderings based on earlier conflicts,
3. *Heuristic Backjumping* (HB) a scheme that gives up searching areas of the search space that require too much work.

Related work in scheduling includes that of Prosser and Burke who use N-th order shallow learning to solve one-machine scheduling problems [3], and that of Badie et al. whose system implements a variation of deep learning in which a minimum set is heuristically selected as the culprit [1].

The remainder of this paper is organized as follows. Section 2 provides a more formal definition of the job shop CSP. Section 3 describes the backtrack search procedure considered in this study. Sections 4, 5 and 6 successively describe each of the three backtracking

[1] A search state is associated with each partial solution. Each search state defines a new CSP whose variables are the variables that have not yet been instantiated and whose constraints are the initial problem constraints along with constraints reflecting current assignments.

schemes developed for this study. Experimental results are presented in section 7. Section 8 summarizes the contributions of this paper.

2 Job Shop Constraint Satisfaction Problem and Search Procedure

The job shop scheduling problem requires scheduling a set of jobs $J = \{j_1, ..., j_n\}$ on a set of physical resources $RES = \{R_1, ..., R_m\}$. Each job j_l consists of a set of operations $O^l = \{O^l_1, ..., O^l_{n_l}\}$ to be scheduled according to a process routing that specifies a partial ordering among these operations (e.g. O^l_i BEFORE O^l_j).

In the job shop CSP studied in this paper, each job j_l has a release date rd_l and a due-date dd_l between which all its operations have to be performed. Each operation O^l_i has a fixed duration du^l_i and a variable start time st^l_i. The domain of possible start times of each operation is initially constrained by the release and due dates of the job to which the operation belongs. If necessary, the model allows for additional unary constraints that further restrict the set of admissible start times of each operation, thereby defining one or several time windows within which an operation has to be carried out (e.g. one or several shifts in factory scheduling). In order to be successfully executed, each operation O^l_i requires p^l_i different resources (e.g. a milling machine and a machinist) R^l_{ij} $(1 \leq j \leq p^l_i)$, for each of which there may be a pool of physical resources from which to choose, $\Omega^l_{ij} = \{r^l_{ij1}, ..., r^l_{ijq^l_{ij}}\}$, with $r^l_{ijk} \in RES (1 \leq k \leq q^l_{ij})$ (e.g. several possible milling machines).

More formally, the problem can be defined as follows:

VARIABLES:

A vector of variables is associated with each operation, $O^l_i (1 \leq l \leq n, 1 \leq i \leq n_l)$, which includes:

1. the *operation start time*, st^l_i, and
2. each *resource requirement*, $R^l_{ij} (1 \leq j \leq p^l_i)$ for which the operation has several alternatives.

CONSTRAINTS:

The non-unary constraints of the problem are of two types:

1. *Precedence constraints* defined by the process routings translate into linear inequalities of the type: $st^l_i + du^l_i \leq st^l_j$ (i.e. O^l_i BEFORE O^l_j);
2. *Capacity constraints* that restrict the use of each resource to only one operation at a time translate into disjunctive constraints of the form: $(\forall p \forall q R^k_{ip} \neq R^l_{jq}) \vee st^k_i + du^k_i \leq st^l_j \vee st^l_j + du^l_j \leq st^k_i$. These constraints simply express that, unless they use different resources, two operations O^k_i and O^l_j cannot overlap [2].

Additionally, there are unary constraints restricting the set of possible values of individual variables. These constraints include non-relaxable due dates and release dates, between which all operations in a job need to be performed. The model can actually accommodate any type of unary constraint that further restricts the set of possible start times of an operation. Time is assumed discrete, i.e. operation start times and end times can only take integer values. Finally, each resource requirement R^l_{ij} has to be selected from a set of resource alternatives, $\Omega^l_{ij} \subseteq RES$.

OBJECTIVE:

In the job shop CSP studied in this paper, the objective is to come up with a feasible solution as fast as possible. Notice that this objective is different from simply minimizing the number of search states visited. It also accounts for the time spent by the system deciding which search state to explore next.

3 The Search Procedure

A depth-first backtrack search procedure is assumed, in which search is interleaved with the application of consistency enforcing mechanisms and variable/value ordering heuristics that attempt to steer clear of deadend states. Search proceeds according to the following steps:

1. If all operations have been scheduled then stop, else go on to 2;
2. Apply the *consistency enforcing* procedure;
3. If a deadend is detected then *backtrack* (i.e. select an alternative if there is one left and go back to 1, else stop and report that the problem is infeasible), else go on to step 4;
4. Select the next operation to be scheduled (*variable ordering* heuristic);
5. Select a promising reservation for that operation (*value ordering* heuristic);
6. Create a *new search state* by adding the new reservation assignment to the current partial schedule. Go back to 1.

The default consistency enforcing scheme and variable/value ordering heuristics used in the procedure are the ones described in [20]:

Consistency Enforcement: The consistency enforcing procedure is a hybrid procedure that differentiates between precedence constraints and capacity constraints. It guarantees that backtracking only occurs

[2] These constraints have to be generalized when dealing with resources of capacity larger than one.

as the result of capacity constraint violations. Essentially, consistency with respect to precedence constraints is enforced by updating in each search state a pair of earliest/latest possible start times for each unscheduled operation. Consistency enforcement with respect to capacity constraints tends to be significantly more expensive due to the disjunctive nature of these constraints. For capacity constraints, a forward checking type of consistency checking is generally carried out by the system. Whenever a resource is allocated to an operation over some time interval, the forward checking procedure checks the set of remaining possible start times of other operations requiring that resource, and removes those start times that would conflict with the new assignment. The system further checks for consistency with respect to a set of redundant capacity constraints, which can be quickly enforced in each search state. This includes checking that no two unscheduled operations totally rely on the same resource over overlapping time intervals [3].

Variable/Value Ordering Heuristics: The default variable/value ordering heuristics used by the search procedure are the *Operation Resource Reliance* (ORR) variable ordering heuristic and *Filtered Survivable Schedules* value ordering heuristic described in [20]. The ORR variable ordering heuristic aims at reducing backtracking by first scheduling difficult operations, namely operations whose resource requirements are expected to conflict with the resource requirements of other operations. The FSS value ordering heuristic is a least constraining value ordering heuristic. It attempts to further reduce backtracking by assigning reservations that are expected to be compatible with a large number of schedules.

These default consistency enforcing schemes and variable/value ordering heuristics have been reported to outperform several other schemes described in the literature, both generic CSP heuristics and specialized heuristics designed for similar scheduling problems [20, 19]. These are efficient schemes that seem to provide a good compromise between the efforts spent enforcing consistency, ordering variables, or ranking assignments for a variable and the actual savings obtained in search time. Nevertheless, because the job shop CSP is an NP-complete problem, these procedures are not sufficient to guarantee backtrack-free search.

The remainder of this paper describes new backtracking schemes that help the system recover from deadend states. It will be seen that, when the default consistency enforcing scheme and/or variable ordering scheme are not sufficient to stay clear of deadends, look-back mechanisms can be devised that will modify these schemes so as to avoid repeating past mistakes (i.e.so as to avoid reaching similar deadend states).

[3] See [20] for further details.

4 Dynamic Consistency Enforcement (DCE)

Backtracking is generally an indication that the default consistency enforcing scheme and/or variable/value ordering heuristics used by the search procedure are insufficient to deal with the subproblems at hand. For this reason, when it reaches a deadend, the system will generally start thrashing if it keeps on using the same default mechanisms[4]. Theoretically, thrashing could be eliminated by enforcing full consistency in each search state. Clearly such an approach is prohibitively expensive. Instead, if one could heuristically identify small subproblems that are likely to be at the source of the conflict and just check for consistency among the variables in these subproblems, thrashing could often be eliminated at a lower computational cost. This is the approach described in this section. A backtracking scheme called *Dynamic Consistency Enforcement* (DCE) is presented that dynamically identifies small critical resource subproblems expected to be at the source of the current deadend. Experimental results reported in Section 7 suggest that, by *selectively* checking for consistency with respect to capacity constraints among the operations in these small subproblems, this scheme is often able to quickly recover from deadends.

When a deadend is detected, DCE checks for consistency with respect to capacity constraints in critical resource subproblems, in order to approximate the full extent of the current deadend and decide how far to backtrack. The critical subproblems used by DCE consist of groups of operations participating in the current conflict along with groups of critical operations identified at an earlier stage. Below, we refer to the group(s) of operations participating in the current conflict, as the *Partial Conflicting Set* of operations (PCS): these are the operations identified by the default consistency enforcing mechanism as having no possible reservations left in the current search state. The objective of the backtracking scheme is to identify the most recent assignment(s), which, if undone, will produce a consistent search state, i.e. a search state in which operations in PCS have reservations that do not seem to conflict with earlier assignments. To this end, DCE checks for consistency with respect to capacity constraints between operations in PCS and critical operations in a so-called *Dangerous Group* (DG) of operations identified earlier. At each level (while backtracking), the set consisting of the union of the PCS, the DG and the set of undone operations up to that level is referred to as the *Deadend Operation Set* (DOS). While backrtacking, DCE performs full consistency checking with re-

[4] Experiments reported in [20, 19] consistency displayed a dual behavior: the vast majority of the scheduling problems were either solved *without* backtracking whatsoever, or required an *exponential amount of chronological backtracking.*

spect to capacity constraints among operations in the DOS. Generally, because the DOS may contain operations requiring different resources, the backtracking scheme checks for consistency with respect to capacity constraints in several resource subproblems [5]. During backtracking the PCS and the DG remain the same and the DOS varies as more undone operations are unioned. At the end of a backtracking episode, DOS has maximum size, call it DOS_{max}. Assuming that the procedure was able to backtrack to a consistent search state [6], DOS_{max} contains all the operations at the origin of the deadend (and often more). DOS_{max} is then saved for later use in a data structure referred to as the *Former Dangerous Groups* (FDG). Details regarding the management of this data structure are provided in subsection 4.1. If a related backtracking episode is later encountered by the system, DOS_{max} can then be retrieved and serve as the DG for this new episode [7]. If a subsequent backtracking episode is unrelated to any of the previous ones, then the DG for this episode is empty.

The behavior of the DCE procedure is illustrated in Figure 1. Each node represents a search state, labeled by the operation that was last scheduled to reach that state, the resource allocated to that operation, and the operation's start time. In this example, search is assumed to have reached a deadend at depth D_5. Operations in the PCS are those operations whose domains of possible start times were identified as empty at depth D_5 due to capacity constraint violations. The resources associated with operations in the PCS are called the *critical resources*. Although a PCS can in general contain operations associated with more than one critical resources, it is often the case that the operations in PCS require the same resource (i.e., the deadend happened as a result of capacity constraint violations on a single resource). Upon encountering a deadend at D_5, DCE backtracks to D_4 and performs full consistency checking with respect to capacity constraints on the set of operations $DOS_4 = PCS \cup DG \cup O_m$ If there are still capacity constraint violations at D_4, operation O_l is undone, and full consistency checking is performed on the new DOS, namely $DOS_3 = PCS \cup DG \cup O_l, O_m$ The procedure is repeated until a consistent DOS is found ($DOS_{max} = DOS_1$ in this example). At this point, the DOS_{max} is saved to be used in the DG for the next related backtracking episode.

4.1 Management of Dangerous Groups

The purpose of the *Former Dangerous Groups* of operations (FDG) maintained by the system is to help determine more efficiently and more precisely the scope of each deadend by focusing on critical resource subproblems. Each group of operations in the FDG consists of operations that are in high contention for the allocation of a same resource. Accordingly, whenever, a conflict is detected that involves some of the operations in one group, the backtracking procedure checks for consistency among *all* operations in that group.

The groups of operations in the FDG are built from the *Deadend Operation Sets* (DOS) obtained at the end of previous backtracking episodes (DOS_{max}). Indeed, whenever a backtracking episode is completed, DOS_{max} is expected to contain all the conflicting operations at the origin of this episode. Generally, DOS_{max} may involve one or several resource subproblems (i.e. groups of operations requiring the same resource). Each one of these subproblems is merged with *related* subproblems currently stored in the FDG. If there is no related group in FDG, the new group is separately added to the data structure.

As operations are scheduled, they are removed from the FDG.

4.2 Additional "Watch Dog" Consistency Checks

Because groups of operations in the FDG are likely deadend candidates, our system further performs simple "watch dog" consistency checks on these dynamic groups of operations. More specifically, for each group G of operations in FDG, the system performs a rough consistency check to see if the resource can still accommodate all the operations in the group. This is done using redundant constraints of the form: $Max(lst_i^l + du_i^l, O_i^l \in G) - Min(est_i^l, O_i^l \in G) \geq \sum_{O_i^l \in G} du_i^l$ where est_i^l and lst_i^l are respectively the earliest and latest possible start times of O_i^l in the current search state.

Whenever such a constraint is violated, an inconsistency has been detected. Though very simple and inexpensive, these checks enable to catch inconsistencies involving large groups of operations that would not be immediately detected by the default consistency mechanisms [8]. Clearly, some inconsistencies can still escape these rough checks.

[5] Because full consistency checking is expensive, if this set is too large, two approaches can be taken to limit computational cost: (1) full consistency checking can be performed only for a subset of the DOS, or (2) k-consistency [10] can be performed, where k is some predetermined number.

[6] Clearly, there is no guarantee that the search state in which DCE stops backtracking is a consistent search state. Experimental results suggest however that this is often the case.

[7] Two backtracking episodes are defined to be related if they are due to capacity constraint violations on *the same resource and over close time intervals*. Otherwise, they are unrelated.

[8] Notice that, when a "watch dog" check fails, PCS is empty.

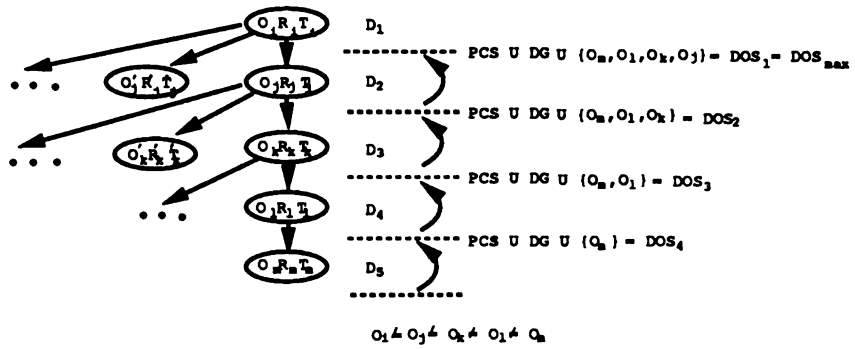


Figure 1: The DCE Backtracking Scheme.

5 Learning From Failures (LFF)

Encounter of a deadend is also often an indication that the default variable ordering was not adequate for dealing with the subproblem at hand. Typically the operations participating in the deadend turn out to be more difficult than the operations selected by the default variable ordering heuristic. It is therefore a good idea to first schedule the operations participating in the conflict that was just resolved. *Learning From Failure* (LFF) is an adaptive procedure that overrides the default variable ordering in the presence of conflicts.

After recovering from a deadend (i.e. after backtracking all the way to an apparently consistent search state), LFF uses the Partial Conflicting Set (PCS) of the deadend to reorganize the ordering in which operations will be rescheduled and make sure that operations in the PCS are scheduled first. This is done using a quasi-stack on which operations in the PCS are pushed in descending order of domain size (operations with more available start times go first)[9]. This orders operations in terms of their criticality (most critical operation on top) so as to ensure that, as S is popped, the most critical operations will be scheduled first. As long as S is non-empty, operations from S are popped and successively scheduled, thus overriding the default variable ordering.

6 A Backjumping Heuristic

Traditional
backtrack search procedures only undo decisions that have been proven to be wrong/inconsistent. Proving that an assignment is inconsistent with others can be very expensive, especially when dealing with large conflicts. Graph-based backjumping and N-th order shallow/deep learning attempt to reduce the complexity of fullblown dependency-directed backtracking by either simplifying the process of identifying inconsistent decisions (e.g. based on the topology of the constraint graph) or restricting the size of the conflicts that can be detected. The Dynamic Consistency Enforcement (DCE) procedure described in Section 6 also aims at reducing the complexity of identifying the source of a conflict by dynamically focusing its effort on small critical subproblems. None of these techniques can be expected to perform well when dealing with large complex [10] conflicts, either because they are too expensive to run or because they deliberately overlook large conflicts. Large complex conflicts can force the search procedure to thrash, even when using procedures such as DCE. In these situations, it may be worth undoing decisions that are not provably wrong but simply appear overly restrictive. Clearly, the resulting search procedure is no longer complete and may fail to find solutions to feasible problems.

Texture measures such as the ones described in [9] could be used to estimate the tightness of different search states, for instance, by estimating the number of global solutions compatible with each search state [11]. Assignments leading to much tighter search states would be prime candidates to be undone. The *Backjumping Heuristic* (BH) used in this study is simpler and, yet, often seems to get the job done. Whenever the system starts thrashing, this heuristic backjumps all the way to the first search state and simply tries the next best value (i.e. reservation) for the critical operation in that state (i.e. the first operation selected by the variable ordering heuristic). BH considers that the search procedure is thrashing when more than θ assignments had to be undone since the procedure began or since the last time the system was thrashing,

[9] If a candidate operation is already on S, i.e. it is encountered for a second time, it is pushed again as though it had a smaller domain.

[10] There are conflicts involving large numbers of variables that are easy to catch, as illustrated by the watch dog checks described in Section 4.

[11] A search state whose partial solution is compatible with a large number of global solutions is a loosely constrained search state, whereas one compatible with a small number of global solutions is tightly constrained.

where θ is a parameter of the search procedure.

7 Experimental Results

Two sets of 40 scheduling problems each were generated that differed in the number of major bottlenecks (one and two major bottlenecks respectively). Each problem had 50 operations and 5 resources (i.e., 10 jobs). All jobs were released at the same time and had to be completed by the same due date. In each problem, the common due date was set so that all operations had to be scheduled within a rather tight estimate of the problem makespan (see [20] for details). These are the conditions in which the default variable/value ordering and consistency enforcing schemes work least effectively (see study reported in [20]). Among these 80 problems, we only report performance on problems in which the default schemes were not sufficient to guarantee backtrack-free search [12]. This leaves 16 scheduling problems with one bottleneck, and 15 with two bottlenecks.

We successively report the results of two studies. The first study compares the performance of three complete backtrack schemes: chronological backtracking, 2d-order deep learning, and the procedure combining the DCE and LFF backtrack schemes described in Section 4 and 5. The second study compares a complete search procedure using the DCE and LFF schemes with an incomplete search procedure combining DCE and LFF with the Backjumping Heuristic (BH) described in Section 6.

7.1 Comparison of Complete Search Procedures

The two intelligent backtracking techniques, DCE and LFF are complementary and were used in combination, denoted by DCE & LFF, for experimentation to assess performance[13]. Each of the problems in the experiment set was run using chronological backtracking, 2d-order deep learning [6] and the DCE & LFF procedures advocated in Section 4 and 5. The results reported here were obtained using a search limit of 500 nodes and a time limit of 1800 seconds (except for deep learning, for which the time limit was increased to 36,000 seconds [14]). All CPU times reported below were obtained on a DECstation 5000 running Knowledge Craft on top of Allegro Common Lisp [15].

Results for the one-bottleneck problems are reported in Table 1. Chronological backtracking solved only 4 problems out of 16. Interestingly enough, deep learning showed no improvement over chronological backtracking either in the number of problems solved or in CPU time. As a matter of fact, deep learning was even too slow to find solutions to some of the problems solved by chronological backtracking. This is attributed to the fact that the constraints in job shop scheduling are more tightly interacting than those in the zebra problem, where the improvement of deep learning over naive backtracking was originally ascertained. On the other hand, DCE & LFF solved 10 problems out of 16 (2 out of these 10 problems were successfully proven infeasible). As expected, by focusing on a small number of critical subproblems, DCE & LFF is able to discover larger more useful conflicts than 2d-order deep learning, while requiring only a fraction of the time required by deep learning. Another observation is that DCE & LFF expanded fewer search states than chronological backtracking for the problems that chronological backtracking solved. However, each of the DCE & LFF expansions took slightly more CPU time, due to the higher level of consistency enforcement.

Results for the set of two-bottleneck problems are reported in Table 2. Similar results are observed here again: deep learning shows no improvement over chronological backtracking and seems significantly slower. The difference between chronological backtracking and DCE&LFF is not as impressive as in the first set of experiments. This is probably because both bottlenecks may have capacity conflicts at the same time. DCE & LFF may then have problems determining which one to consider first. As can be seen from Table 2, chronological backtracking solved 7 out of 15 problems, whereas DCE & LFF solved 8 out of 15. On the problems solved by both chronological backtracking and DCE & LFF, DCE & LFF turned out to be slightly faster overall.

7.2 Complete vs. Incomplete Search Procedures

Table 3 and 4 compare the performance of the complete search procedure using DCE & LFF against that of an incomplete search procedure using DCE & LFF in combination with the Backjumping Heuristic (BH) described in Section 6. While DCE & LFF was able to solve only 10 out of 16 one-bottleneck problems and 8 out 15 two-bottleneck problems, DCE & LFF combined with BH solved 14 one-bottleneck problems and 13 two-bottleneck problems. The only one-bottleneck

[12] Clearly, performance in the absence of backtracking is uninteresting, since our backtracking schemes would never be invoked, i.e. CPU time remains unchanged.

[13] Besides the experiments reported below, additional experiments were performed to assess the benefits of using DCE and LFF separately. These experiments show that both techniques contribute to the improvements reported in this section.

[14] This was motivated by the fact that our implementation of deep learning may not be optimal.

[15] Comparison between C programs and Knowledge Craft programs suggests that the code would run 10 to 20 times faster in C.

Table 1: Results of One-bottleneck Experiments.

Exp. No.	Chronological Backtracking			DCE & LFF			Deep Learning		
	No. of Nodes	CPU (sec)	Result	No. of Nodes	CPU (sec)	Result	No. of Nodes	CPU (sec	Result
1	500	1427	F	122	1232	S*	500	5756	F
2	500	1587	F	500	1272	F	500	5834	F
3	74	148	S	63	117	S	25	36000	F
4	69	152	S	52	120	S	69	391	S
5	500	1407	F	65	134	S	500	11762	F
6	500	1469	F	500	1486	F	500	8789	F
7	500	1555	F	59	130	S	500	9681	F
8	500	1705	F	41	145	S*	500	9560	F
9	53	108	S	53	102	S	53	122	S
10	500	1529	F	500	1536	F	500	9114	F
11	500	1460	F	85	1800	F	500	14611	F
12	500	1694	F	500	1131	F	500	21283	F
13	51	109	S	51	81	S	51	88	S
14	500	1762	F	63	138	S	500	18934	F
15	500	1798	F	69	142	S	500	9601	F
16	500	1584	F	500	1183	F	65	36000	F

S: Solved ; F: Failure; S*: Proved infeasible
Time Limit: 1800 sec (Except Deep Learning)
Node Limit: 500

Table 2: Results of Two-bottleneck Experiments

Exp. No.	Chronological Backtracking			DCE & LFF			Deep Learning		
	No. of Nodes	CPU (sec)	Result	No. of Nodes	CPU (sec)	Result	No. of Nodes	CPU (sec)	Result
1	500	1139	F	113	1800	F	18	36000	F
2	500	1444	F	425	1800	F	115	36000	F
3	84	175	S	109	202	S	84	811	S
4	56	123	S	56	112	S	56	213	S
5	51	101	S	51	113	S	13	36000	F
6	500	1531	F	321	1800	F	328	36000	F
7	500	1775	F	500	1357	F	500	2793	F
8	52	102	S	52	115	S	33	36000	F
9	500	1634	F	247	974	S	500	1519	F
10	500	1676	F	91	1800	F	26	36000	F
11	66	163	S	59	104	S	66	2240	S
12	56	139	S	58	104	S	58	281	S
13	54	129	S	52	91	S	54	28900	S
14	500	1676	F	346	1800	F	500	903	F
15	500	1522	F	324	1800	F	296	36000	F

S: Solved ; F: Failure; S*: Proved infeasible
Time Limit : 1800 sec. (36000 sec. for Deep Learning)
Node Limit : 500

problems that were not solved by DCE & LFF & BH are the two infeasible problems identified by the complete search procedure DCE & LFF. This is hardly a surprise. While the addition of BH to DCE & LFF enables the search procedure to solve a larger number of problems, it also makes the procedure incomplete (i.e. infeasible problems can no longer be identified). Additional experiments combining BH with a simple chronological backtracking scheme also indicate that both DCE & LFF and BH contribute to the good performance of DCE & LFF & BH. Results on two-bottleneck problems (See Table 4) suggest that the impact of the backjumping heuristic is particularly effective on these problems. This is attributed to the fact that two-bottleneck problems give rise to more complex conflicts. Identifying the assignments participating in these more complex conflicts may simply be too difficult for any exact backtracking scheme to work. Instead, because it can undo assignments that are not provably wrong but simply appear overly restrictive, BH seems more effective at dealing with these more complex conflicts.

8 Concluding Remarks

We have presented three intelligent backtracking schemes for the job shop scheduling CSP:

1. *Dynamic Consistency Enforcement* (DCE), a dependency-directed scheme, that dynamically focuses its effort on small critical subproblems,
2. *Learning From Failure* (LFF), which modifies the order in which variables are instantiated based on earlier conflicts, and
3. a *Backjumping Heuristic* which, when thrashing occurs, can undo assignments that are not provably inconsistent but appear overly restrictive.

Table 3: Results of One-bottleneck Experiments.

Exp. No.	DCE & LFF			DCE & LFF & BH		
	No. of Nodes	CPU (sec)	Result	No. of Nodes	CPU (sec)	Result
1	122	1232	S*	350	1800	F
2	500	1272	F	203	1124	S
3	63	117	S	63	123	S
4	52	120	S	52	116	S
5	65	134	S	65	144	S
6	500	1486	F	127	424	S
7	59	130	S	59	125	S
8	41	145	S*	457	1800	F
9	53	108	S	53	100	S
10	500	1536	F	67	170	S
11	85	1800	F	74	170	S
12	500	1131	F	164	616	S
13	51	81	S	51	92	S
14	63	138	S	63	149	S
15	69	142	S	69	158	S
16	500	1183	F	156	524	S

S: Solved ; F: Failure; S*: Proved infeasible
Time Limit: 1800 sec. Node Limit: 500

Table 4: Results of Two-bottleneck Experiments

Exp. No.	DCE & LFF			DCE & LFF & BH		
	No. of Nodes	CPU (sec)	Result	No. of Nodes	CPU (sec)	Result
1	113	1800	F	151	456	S
2	425	1800	F	371	1780	S
3	109	202	S	95	210	S
4	56	112	S	56	108	S
5	51	113	S	51	97	S
6	321	1800	F	420	1800	F
7	500	1357	F	159	534	S
8	52	115	S	52	96	S
9	247	974	S	423	1705	S
10	91	1800	F	440	1800	F
11	59	104	S	59	113	S
12	58	104	S	58	112	S
13	52	91	S	52	102	S
14	346	1800	F	239	512	S
15	324	1800	F	73	195	S

S: Solved ; F: Failure; S*: Proved infeasible
Time Limit: 1800 sec. Node Limit: 500

The significance of this research is twofold:

1. Job shop scheduling problems with non-relaxable time windows have multiple applications, including both manufacturing and space-related applications. We have shown that our schemes combined with powerful techniques that we had previously developed (1) further reduce the average complexity of backtrack search, and (2) enable our system to efficiently solve problems that could not be solved otherwise due to excessive computational cost. While the results reported in this study were obtained on problems that require finding a feasible schedule, the backtracking schemes presented in this paper can also be used on optimization versions of the scheduling problem.
2. This research also points to the deficiencies of dependency-directed backtracking schemes advocated earlier in the literature. In particular, comparison with N-th order deep learning indicates that this technique failed (in our set of experiments) to improve performance when applied to job shop scheduling problems. This is because N-th order deep learning uses constraint size as the only criterion to decide whether or not to record earlier failures. When deep learning limits itself to small-size conflicts, it fails to record some important constraints; when it considers conflicts of larger size, its computational complexity becomes prohibitive. Traditional backtracking schemes never undo assignments unless they can prove that they are at the source of the conflict. When dealing with large complex conflicts, proving that a particular assignment should be undone can be very expensive. Instead, our experiments suggest that, when thrashing cannot easily be avoided, it is often a better idea to use backjumping heuristics that undo decisions simply because they *appear* overly restrictive. When using such heuristics, search completeness can no longer be guaranteed.

Acknowledgements

This research was supported, in part, by the Defense Advanced Research Projects Agency under contract F30602-91-C-0016 and, in part, by the Robotics Institute at Carnegie Mellon University.

References

[1] C. Badie, G. Bel, E. Bensana, and G. Verfaillie. Operations research and artificial intelligence cooperation to solve scheduling problems. In *First International Conference on Expert Planning Systems*, 1990.

[2] J.R. Bitner and E.M. Reingold. Backtrack programming techniques. *Communications of the ACM*, 18(11):651–655, 1975.

[3] Peter Burke and Patrick Prosser. A distributed asynchronous system for predictive and reactive scheduling. Technical Report AISL-42, Department of Computer Science, University of Strathclyde, 26 Richmond Street, Glasgow, GI IXH, United Kingdom, October 1989.

[4] Rina Dechter. Learning while searching in constraint satisfaction problems. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 178–183, 1986.

[5] Rina Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41:273–312, 1989.

[6] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 1989.

[7] Rina Dechter and Judea Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34(1):1–38, 1988.

[8] John Doyle. A truth maintenance system. *Artificial Intelligence*, 12(3):231–272, 1979.

[9] Mark S. Fox, Norman Sadeh, and Can Baykan. Constrained heuristic search. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 309–315, 1989.

[10] E.C. Freuder. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29(1):24–32, 1982.

[11] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Co., 1979.

[12] John Gaschnig. Performance measurement and analysis of certain search algorithms. Technical Report CMU-CS-79-124, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, 1979.

[13] Solomon W. Golomb and Leonard D. Baumert. Backtrack programming. *Journal of the Association for Computing Machinery*, 12(4):516–524, 1965.

[14] Robert M. Haralick and Gordon L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14(3):263–313, 1980.

[15] A.K. Mackworth and E.C. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25(1):65–74, 1985.

[16] Jr. Paul W. Purdom. Search rearrangement backtracking and polynomial average time. *Artificial Intelligence*, 21:117–133, 1983.

[17] N. Sadeh and M.S. Fox. Preference propagation in temporal/capacity constraint graphs. Technical Report CMU-CS-88-193, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, 1988. Also appears as Robotics Institute technical report CMU-RI-TR-89-2.

[18] N. Sadeh and M.S. Fox. Focus of attention in an activity-based scheduler. In *Proceedings of the NASA Conference on Space Telerobotics*, January 1989.

[19] N. Sadeh and M.S. Fox. Variable and value ordering heuristics for hard constraint satisfaction problems: an application to job shop scheduling. Technical Report CMU-RI-TR-91-23, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1992.

[20] Norman Sadeh. *Look-ahead Techniques for Micro-opportunistic Job Shop Scheduling*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, March 1991.

[21] Norman Sadeh and Mark S. Fox. Variable and value ordering heuristics for activity-based job-shop scheduling. In *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, Hilton Head Island, S.C.*, pages 134–144, 1990.

[22] R. Stallman and G. Sussman. Forward reasoning and dependency-directed backtracking in a sysem for computer-aided circuit analysis. *Artificial Intelligence*, 9:135–196, 1977.

[23] K. Sycara, S. Roth, N. Sadeh, and M. Fox. Distributed constrained heuristic search. *IEEE Transactions on System, Man and Cybernetics*, 21(6), 1991.

[24] R.J. Walker. *An Enumerative Technique for a Class of Combinatorial Problems*, volume 10, chapter 7, pages 91–94. American Mathematical Society, Rhode Island, 1960.

Dense Time and Temporal Constraints With ≠

Manolis Koubarakis*
Dept. of Computer Science
University of Toronto
10, King's College Road
Toronto, Ontario M5S 1A4
CANADA

Abstract

In some temporal reasoning systems, inequations can give rise to disjunctions of inequations when variable elimination is performed. Motivated by this observation, we extend previous research on temporal constraints by considering disjunctions of inequations (under the assumption that time is dense.) We present results on consistency checking, canonical forms and variable elimination for this new class of temporal constraints.

1 INTRODUCTION

In recent years temporal reasoning has received much attention from the artificial intelligence and database community (see [SG88, Sno90] for surveys). This is a rather natural trend since reasoning about time is essential in many applications (e.g., natural language understanding, planning, scheduling etc.). In the artificial intelligence community in particular, some researchers have introduced binary constraint networks as reasoning tools for problems involving temporal constraints [All83, VKvB89, LM88, DMP91, KL91, Mei91]. Much of this work has its origins in previous research on general constraint satisfaction problems [Mon74, Mac77, DP88]. We follow this line of work and extend it by considering a more expressive class of temporal constraints.

In our work, time is assumed to be linear, dense and unbounded. Points are the only time entities. Temporal information is represented in terms of *inequalities* and *disjunctions of inequations*. Two kinds of inequalities are allowed: (i) bounds on the distance between two time points e.g., $t_1 - t_2 \leq 5$, and (ii) bounds on the position of a time point on the time-line e.g., $t_1 \geq 5$. *Inequations* are constraints of the form $\alpha \neq \beta$. Similarly with inequalities, two kinds of inequations are allowed: $t_1 - t_2 \neq r$ or $t \neq r$. Although inequations have been considered in the past in the context of binary constraint networks (e.g., by [VKvB89, Mei91]), disjunctions of inequations have been neglected. However, if one considers temporal formalisms or systems with more expressive representation/query languages (e.g., TMM [DM87], Telos [MBJK90] and the model of [Kou92b]), then one cannot deal with binary inequation constraints and avoid non-binary ones. In such systems there are queries that can only be answered by eliminating variables from a set of temporal constraints. But when variable elimination is performed, an inequation can give rise to a disjunction of inequations.[1]

The basic properties of disjunctions of linear inequations have already been studied by [LM89] who considered generalized linear constraints (i.e., linear inequalities and disjunctions of linear inequations). [LM89] observed that disjunctions of inequations can be treated independently of one another for deciding consistency. An important consequence of this fact is that one can decide in polynomial time whether a set of temporal constraints is consistent. Building on the results of [LM89], we were able to make the following contributions:

- We applied the results of [LM89] to the case of temporal constraints (section 3). In particular, we developed algorithms for deciding consistency and computing canonical forms for sets of temporal constraints.
- We developed a variable elimination algorithm for temporal constraints (section 4). This algorithm can also be applied to the more expressive class of generalized linear constraints.
- We showed that the presence of disjunctions of inequations makes variable elimination intractable (theorem 4.4). However, for the limited class of simple temporal constraints (which corresponds to the full point algebra of [VKvB89]) variable

*E-mail: koubarak@cs.toronto.edu

[1] Van Beek makes a very similar observation in the context of decomposable point networks [vB90a].

elimination is tractable (corollary 5.1). The proof of this result is indirect and demonstrates that a decomposable constraint set equivalent to a given set of simple temporal constraints can be found in polynomial time (theorem 5.1). The decomposability of sets of simple temporal constraints has been an open problem since [vB90a].

The organization of the paper is as follows. Section 2 introduces temporal constraints. Section 3 discusses consistency checking and canonical forms for temporal constraints. Section 4 presents our results on variable elimination. Section 5 discusses variable elimination for simple temporal constraints. Section 6 summarizes our contributions and presents some directions for future research. Finally, appendix A contains the proofs of the most important results.

2 TEMPORAL CONSTRAINTS

At first, we present a language $\mathcal{L}$ for talking about time. We consider time to be linear, dense and unbounded. *Points* will be our only time entities. *Interval* information will be represented in terms of endpoint information. Points are identified with the rational numbers but our results still hold if points are identified with the reals.

We begin by giving a formal definition of $\mathcal{L}$. $\mathcal{L}$ is a first order calculus with equality. The logical symbols of $\mathcal{L}$ include parentheses, quantifiers, sentential connectives, a countably infinite set of variables V and $=$ (the equality symbol). The non-logical symbols of $\mathcal{L}$ include a countably infinite set of constants (the rational numerals), a function symbol $-$ of arity 2 and a predicate symbol $<$ of arity 2. The symbols of $\mathcal{L}$ are interpreted with respect to the structure $\mathcal{T}$ which embodies our assumptions for time. $\mathcal{T}$ assigns to each constant a member of the set $\mathbf{Q}$ (i.e., a rational number). To function symbol $-$, $\mathcal{T}$ assigns the function $-_{\mathbf{Q}}$ which is the subtraction operation over the rationals. To predicate symbol $<$, $\mathcal{T}$ assigns the relation $<_{\mathbf{Q}}$ ("less than") over the rationals.

A *variable assignment* v is a function $v : V \rightarrow \mathbf{Q}$ where V is the set of variables of $\mathcal{L}$. We use $\mathcal{T} \models \phi[v]$ to denote that the structure $\mathcal{T}$ *satisfies* ϕ with variable assignment v. Similarly, $\mathcal{T} \models \sigma$ denotes that the sentence σ is *true* in the structure $\mathcal{T}$. We will also write $\phi \models \psi$ whenever $\mathcal{T} \models \phi \supset \psi$. Satisfaction and truth for the structure $\mathcal{T}$ are defined in the usual way [End72].

We will now restrict our attention to certain formulas of $\mathcal{L}$ which will be used to express temporal information. *Temporal constraints* are formulas of $\mathcal{L}$ which fall under any of the following categories:

- *Inequalities*: $t_1 - t_2 \leq r$ where t_1, t_2 are variables and r is a constant (t_1 or t_2 can be omitted).[2]
- *Disjunctions of inequations*:
$$t_1 - t'_1 \neq r_1 \vee \cdots \vee t_n - t'_n \neq r_n$$
where $t_1, \cdots, t_n, t'_1, \cdots, t'_n$ are variables and $r_1, \cdots, r_n$ are constants (t_i or t'_i can be omitted).
- *true* and *false* with the obvious semantics.

Using the above constraints equalities can be written as conjunctions of two inequalities. Similarly, strict inequalities can be written as conjunctions of an inequality and an inequation.

A temporal constraint of the form $t_1 - t_2 \leq r$ or $t_1 - t_2 \neq r$ will be called *atomic*. A temporal constraint of the form $t_1 \leq t_2$ or $t_1 \neq t_2$ is called *simple*. Simple temporal constraints have the same expressive power with the full point algebra of [VKvB89]. If c is a disjunction of inequations then $\overline{c}$ denotes the *complement* of c i.e., the conjunction of equations obtained by negating c.

For a set C of temporal constraints in variables $x_1, \ldots, x_n$ the *solution set* of C is defined as $Sol(C) = \{(\tau_1, \ldots, \tau_n) : \tau_i \in \mathbf{Q}$ and for every $c \in C$, $\mathcal{T} \models c[x_1 \leftarrow \tau_1, \ldots, x_n \leftarrow \tau_n]\}$. Each member of $Sol(C)$ is called a *solution* of C. If C is a set of equalities in n variables, the solution set of C is an affine subset of $\mathbf{Q}^n$. If C is a set of inequalities in n variables, the solution set of C is a convex polyhedron in $\mathbf{Q}^n$. If C is a set of disjunctions of inequations, the solution set of C is $\mathbf{Q}^n \setminus Sol(\{\overline{c} : c \in C\})$.

A set of temporal constraints is called *consistent* or *satisfiable* if and only if $Sol(C) \neq \emptyset$. Two sets of constraints are called *equivalent* if and only if they have the same solution set. A set of constraints is the algebraic counterpart of the logical conjunction of its members. Thus, in the following sections, we will frequently mix the terms "set of constraints" and "conjunction of constraints".

Inequality constraints have so far been studied with the help of constraint networks [LM88, vB90b, DMP91, KL91, Mei91]. A *binary constraint network* is a directed graph whose nodes represent variables and whose arcs represent binary constraints between these variables. For example, if the constraints between variables x_i and x_j are $x_i - x_j \leq 3$ and $x_j - x_i \leq 0$ then arc $j \rightarrow i$ is labeled with 3 and arc $i \rightarrow j$ is labeled with 0. Inequalities with only one variable are accomodated in these networks with the introduction of an auxiliary variable $x_0 = 0$.[3]

The notions of solution, consistency and satisfiability can also be defined for constraint networks in the ob-

[2] For conjunctions of inequalities, we will usually write $\alpha \leq \beta \leq \gamma$ instead of $\alpha \leq \beta \wedge \beta \leq \gamma$.

[3] [DMP91] call this graph the *distance graph* of the constraints. They reserve the term binary constraint network for a different but equivalent representation.

vious way. Two networks are called *equivalent* if they have the same set of solutions. Let us now assume that N and N' are networks with the same set of variables X. If $x_i, x_j \in X$ then $c_N(x_i, x_j)$ will denote the conjunction of the constraints between x_i and x_j in N. We will say that N is *tighter* than N' (denoted $N \subseteq N'$) if and only if for each pair of variables $x_i, x_j \in X$, $c_N(x_i, x_j) \models c_{N'}(x_i, x_j)$. For each network N there is a unique equivalent network N' which is minimal with respect to $\subseteq$. This network is called the *minimal network*. If C is a set of inequalities then $NET_m(C)$ denotes its minimal network representation. If N is a network then $Constraints(N)$ will denote the set of constraints represented by N.

Let us now compare the class of temporal constraints considered in this paper to the classes considered previously by other researchers. Inequalities have been investigated thoroughly in the past [DMP91, VKvB89, Mei91, KL91] but inequations have received less attention. Inequations of the form $t_1 \neq t_2$ have been considered by [vB90a] in the context of point networks. Inequations of the form $t_1 - t_2 \neq r$ or $t \neq r'$ (r, r' are real constants) can be represented in the binary constraint networks of [DMP91] and the generalized temporal constraint networks of [Mei91]. In addition, [Mei91] studies inequations of the form $t \neq r$ (r a real constant) in the context of point networks with *almost-single-interval domains*. To the best of our knowledge, no one has considered disjunctions of inequations so far. Nevertheless, disjunctions of inequations are useful for the representation of several temporal situations as it is demonstrated by the following examples:

- Jobs $J1$ and $J2$ should not both start at the time job J starts: $J_s \neq J1_s \vee J_s \neq J2_s$
- It took robots A and B at most five minutes to finish job J; however, one of the robots worked for a shorter time:
  $A_{end} - A_{begin} \leq 5,\ B_{end} - B_{begin} \leq 5,$
  $A_{end} - A_{begin} \neq 5 \vee B_{end} - B_{begin} \neq 5$
- Intervals i and j are not equal:[4]
  $i_L \neq j_L \vee i_R \neq j_R$
- Let us assume that the following information is given to a natural language understanding system. "David and Murray arrived at the accident scene together. Murray left at least five minutes later than Bill. Bill left at least three minutes later than David. John and Bill did not leave the scene together." If we assume that $scene(p, t_L, t_R)$ represents the fact that person p was at the accident scene for an interval whose endpoints are t_L and t_R then this information can be represented by the following first order logic formulas (in clausal form):
  $scene(david, d_L, d_R),\ scene(murray, m_L, m_R),$
  $scene(john, j_L, j_R),\ scene(bill, b_L, b_R),$
  $d_L < d_R,\ m_L < m_R,\ b_L < b_R,\ j_L < j_R,$
  $d_L = m_L,\ m_R - b_R \geq 5,\ b_R - d_R \geq 3,\ b_R \neq j_R.$
  In these formulas, *david*, *murray*, *john* and *bill* are constants different from each other (unique names assumption [Rei80]) but $d_L, d_R, \ldots, b_L, b_R$ are Skolem constants characterized only by the given constraints.

  Let us now assume that the query "What are the *possible* times that John, David and Murray were at the scene ?" is posed. To answer this query, we have to compute all 6-tuples (J_L, J_R, D_L, D_R, M_L, M_R) which satisfy the following constraints:
  $D_L < D_R,\ M_L < M_R,\ b_L < b_R,\ J_L < J_R,$
  $D_L = M_L,\ M_R - b_R \geq 5,\ b_R - D_R \geq 3,\ b_R \neq J_R.$
  Obviously this answer set is *infinite*. In the temporal deductive database model of [Kou92b] this query will be answered by returning a *finite representation* of the answer set.[5] This representation is computed by *eliminating* b_L and b_R from the above set of constraints to obtain:
  $D_L < D_R,\ M_L < M_R,\ J_L < J_R,\ D_L = M_L,$
  $M_R - D_R \geq 8,\ J_R - D_R \neq 3 \vee J_R - M_R \neq -5.$

The last example is important. It shows that an inequation can give rise to a disjunction of inequations when variable elimination is performed. This fact motivated the work presented in the following sections.

3 DECIDING CONSISTENCY AND COMPUTING CANONICAL FORMS FOR TEMPORAL CONSTRAINTS

In this section, we discuss two problems: (i) deciding whether a set of temporal constraints is consistent, and (ii) computing the first and second canonical form for a consistent set of temporal constraints. Since most of our results are consequences of [LM89], our presentation will be rather terse. The reader can consult [LM89] and [Kou92a] for proofs and more details.

The following result of [LM89] shows that disjunctions of inequations can be treated independently of one another for deciding consistency.

Lemma 3.1 *(Independence of disjunctions of inequations.) Let C_i be a set of inequalities and C_n be a set of disjunctions of inequations. The set $C_i \cup C_n$ is consistent if and only if C_i is consistent and each disjunction of inequations in C_n is consistent with the inequalities in C_i.*

[4] In this and the following example i_L and i_R denote the left and right endpoint of any interval i respectively.

[5] The TMM system [DM87] also offers a limited form of this query answering capability. But TMM's representation language does not support inequation constraints.

This lemma is very important since it allows one to avoid the combinatorial explosion resulting from translating inequations into disjunctions of strict inequalities.

We will now define the first canonical form for temporal constraints. In a temporal reasoning system (e.g., TMM or a system based on the model of [Kou92b]), a canonical form can be useful for answering queries, standardizing output and determining the equivalence of two sets of temporal constraints. In previous work on inequality constraints the role of a canonical form was played by the minimal network representation [DMP91]. Before we proceed, we need a few definitions.

A set of equalities in variables $y_1, \ldots, y_m, x_1, \ldots, x_k$ is in *solved form* if and only if it is

$$\{y_1 = x_{j_1} + r_1,\ y_2 = x_{j_2} + r_2, \ldots, y_m = x_{j_l} + r_m\}$$

where $y_1, \ldots, y_m, x_1, \ldots, x_k$ are distinct variables, $\{x_{j_1}, \ldots, x_{j_l}\} \subseteq \{x_1, \ldots, x_k\}$ and $r_1, \ldots, r_m$ are constants. The variables $y_1, \ldots, y_m$ are called the *eliminable* or *bound* variables and $x_1, \ldots, x_k$ are called the *parameters* of the solved form. A disjunction of inequations is in solved form if and only if its complement is in solved form.

A disjunction of inequations c is called C_i*-precise* (or simply *precise* when C_i is understood) if and only if it is not implied by the inequalities in C_i, and the affine closure of $Sol(C_i) \cap Sol(\bar{c})$ is $Sol(\bar{c})$ (the latter requirement simply says that c is given in its smallest possible dimension).

A constraint c is *redundant* in the set of constraints C if and only if $Sol(C) = Sol(C - \{c\})$.

A set C of temporal constraints in variables $y_1, \ldots, y_m, x_1, \ldots, x_k$ is in *first canonical form* if and only if it does not contain redundant constraints and consists of (i) a set of equalities C_e in variables $y_1, \ldots, y_m, x_1, \ldots, x_k$ in solved form, (ii) a set of inequalities C_i in variables $x_1, \ldots, x_k$ which does not contain any implicit equalities,[6] and (iii) a set of C_i-precise disjunctions of inequations in variables $x_1, \ldots, x_k$ in solved form.

Figures 1 and 2 present an algorithm for computing the first canonical form of a set of temporal constraints. We now discuss this algorithm in detail and calculate its time complexity.

In this and subsequent sections, complexity bounds for our algorithms are calculated in terms of *cardinality*, *size* and *number of variables* of constraint sets. The cardinality of a set of constraints C is denoted by $|C|$. The number of variables in a set of constraints C is denoted by $v(C)$. The size of a constraint set C, which is denoted by $\|C\|$, is the sum of the sizes of its elements.

[6] If C is a set of inequalities then $x_i - x_j \leq r \in C$ is called an *implicit equality* if and only if $C \models x_i - x_j = r$.

Algorithm First_Canonical_Form

Input: A set of temporal constraints $C = C_e \cup C_i \cup C_n$ where C_e is a set of equalities, C_i is a set of inequalities and C_n is a set of disjunctions of inequations.

Output: INCONSISTENT if C is inconsistent, otherwise a constraint set equivalent to C in first canonical form.

Method:
Step 1: Detect possible inconsistencies in $C_e \cup C_i$.
$N := NET_m(C_e \cup C_i)$.
If N contains an inconsistency then
  Return INCONSISTENT
EndIf

Step 2: Find all implicit/explicit equalities of $C_e \cup C_i$ and return them in solved form.
$C_e^1 := Equalities(N)$.

Step 3: Eliminate all bound variables from C_i and C_n.
$C_i^1 := \emptyset$; $C_n^1 := \emptyset$
For every equation $y = x + r$ in C_e^1 do
  For every $c \in C_i$ do
    If y appears in c then
      Substitute $x + r$ for y in c
      Simplify c and add it to C_i^1
    EndIf
  EndFor
  For every $c \in C_n$ do
    If y appears in c then
      Substitute $x + r$ for y in c; Simplify c
      If c evaluates to $0 \neq 0$ then
        Return INCONSISTENT
      Else
        Add c to C_n^1
      EndIf
    EndIf
  EndFor
EndFor

Step 4: Remove redundant inequalities.
Call *Redundant_Inequalities*(C_i^1)

Figure 1: Computing the first canonical form

Algorithm First_Canonical_Form (continued)

Step 5: Determine which disjunctions of inequations are not implied by the inequalities in C_i^1. Transform these constraints into precise ones.

Initialize C_n^2 to $\emptyset$
For each $c \in C_n^1$ do
  Negate c to obtain a set of equalities E
  $N := NET_m(E \cup C_i^1)$
  If there is an inconsistency in N then
    Discard c (c is a tautology or it is already implied by C_i^1)
  Else
    $E' := Equalities(N)$
    Negate the equalities of E' to obtain a precise constraint c. Add c to C_n^2.
  EndIf
EndFor

Step 6: Remove redundancy from C_n^2.
Call *Redundant_Inequations*(C_n^2)

Step 7: Return the normal form of C.
Return $(C_e^1,\ C_i^1,\ C_n^2)$

Figure 2: Computing the first canonical form

The size of a constraint is the number of atomic constraints appearing in its disjunctive normal form. For example, $\|x_1 - x_2 \leq 3\| = 1$, $\|2 \leq x_1 - x_2 \leq 3\| = 2$ and $\|x_1 \neq x_2 \vee x_1 \neq x_3\| = 2$. The following lemma will be useful in calculating complexity bounds. Its proof is an easy consequence of the previous definitions.

Lemma 3.2
1. For every set of constraints C, $v(C) \leq 2\|C\|$.
2. If C is a non-redundant set of inequalities then $|C| \leq v(C)^2 + v(C)$.
3. If C is a set of atomic constraints then $\|C\| = |C|$.
4. If C is a set of disjunctions of inequations in solved form then $\|C\| \leq |C|\, v(C)$.

Initially, the algorithm discovers whether there is an inconsistency in the constraints $C_e \cup C_i$. This step takes $O(v(C_e \cup C_i)^3)$ time. Step 2 computes the solved form of all explicit and implicit equalities in the constraints. This is done by algorithm *Equalities* which can be found in [Kou92a]. This step takes time $O(v(C_e \cup C_i)^3)$ and generates at most $|C_e^1| = O(v(C_e \cup C_i))$ equalities. Step 3 eliminates all bound variables of C_e^1 from $C_i \cup C_n$. This step takes $O(v(C_e \cup C_i)(|C_i| + |C_n|\, v(C_n)))$ time. Step 4 calls algorithm *Redundant_Inequalities* (given in [Kou92a]) to remove redundant inequalities. This step takes $O(|C_i|\, v(C_i)^3)$ time. Step 5 rewrites each disjunction of inequations in its precise form. This step takes $O(|C_n|\,(v(C_i \cup C_n)^3)$ time. Step 6 calls algorithm *Redundant_Inequations* (given in [Kou92a]) removes redundancy from disjunctions of inequations. This step takes $O(|C_n|^2\, v(C_n)^2)$ time.

Using lemma 3.2, we can now conclude that the algorithm takes $O(\|C\|^4)$ time. Thus we have proved the following theorem.

Theorem 3.1 *Deciding consistency and computing the first canonical form of a set of temporal constraints can be done in $O(\|C\|^4)$ time.*

Example 3.1 For the constraint set $\{x_6 - x_3 = 7,\ x_5 \leq x_1,\ x_1 \leq x_5,\ x_5 \leq x_3,\ x_3 \leq x_2,\ x_3 - x_2 \leq 4, x_1 \neq x_2\}$, *First_Canonical_Form* returns $\{x_5 = x_1,\ x_6 = x_3 + 7,\ x_1 - x_3 \leq 0,\ x_3 - x_2 \leq 0,\ x_2 - x_1 \neq 0 \vee x_3 - x_1 \neq 0\}$.

The following uniqueness theorem is a consequence of [LM89]. The assumption underlying the proof of the theorem is that there is a fixed order of variables and this order is taken into account consistently when equations or inequations are transformed into solved form (e.g., in example 3.1 the lowest numbered vertex is always chosen to be a parameter).

Theorem 3.2 *For equivalent sets of constraints in the same set of variables, the algorithm First_Canonical_Form outputs identical canonical forms.*

The next theorem calculates the complexity of deciding whether a constraint is implied by a constraint set in first canonical form.

Theorem 3.3 *Let us assume that the first canonical form of C is (C_e, C_i, C_n). Then the following statements are true. For statements 2 and 3 we assume that x_i and x_j are parameter variables.*
1. $C \models x_i - x_j = r$ if and only if $C_e \models x_i - x_j = r$. This entailment can be checked in $O(|C_e|)$ time.
2. $C \models x_i - x_j \leq r$ if and only if $C_i \models x_i - x_j \leq r$. This entailment can be checked in $O(v(C_i)^3)$ time.
3. If c is a disjunction of inequations, $C \models c$ if and only if $C_i \models c$ or $C_n \models c'$ and c' is the precise form of c. This entailment can be checked in $O(v(C_i)^3 + |C_n|\, v(c)v(C_n))$ time.

Let us now assume that C is a set of inequalities and compare the first canonical form of C with the minimal network $NET_m(C)$. The network $NET_m(C)$ can be built in $O(v(C)^3)$ time. It contains redundant constraints, but this allows us to determine in constant time whether an inequality is implied by C. On the other hand, the first canonical form of C can be built in $O(|C|\, v(C)^3)$. It does not contain redundancy and requires $O(v(C)^3)$ time for answering the same query (theorem 3.3). This suggests that the following combination of the two representations can be advantageous.

A set C of temporal constraints in variables $y_1, \ldots, y_m, x_1, \ldots, x_k$ is in *second canonical form* if and only if it consists of (i) a non-redundant set of equalities C_e in variables $y_1, \ldots, y_m, x_1, \ldots, x_k$ in solved form, (ii) a set of inequalities C_i in variables $x_1, \ldots, x_k$ such that C_i does not contain implicit equalities and $C_i = Constraints(NET_m(C_i))$, and (iii) a set of C_i-precise disjunctions of inequations in variables $x_1, \ldots, x_k$ in solved form. The second canonical form can be built in $O(\|C\|^4)$ time by a variation of the algorithm for the canonical form. In this case, determining whether an inequality is implied by C can be done in constant time. Determining whether an equality or a disjunction of inequations is implied by C is as for the first canonical form.

4 VARIABLE ELIMINATION FOR TEMPORAL CONSTRAINTS

In this section, we present an algorithm for eliminating a variable from a set of temporal constraints. Variable elimination is the algebraic counterpart of the quantifier elimination operation of Mathematical Logic and the projection operation of Geometry. We will say that the variable x_1 has been *eliminated* from a set of constraints C with variables $x_1, x_2, \ldots, x_n$ if C is transformed into a new set of constraints C' such that x_1 does not appear in C' and every solution $(x_2^0, x_3^0, \ldots, x_n^0)$ of the constraints in C' can be extended to a solution $(x_1^0, x_2^0, \ldots, x_n^0)$ of the constraints in C.

Let us first sketch a variable elimination algorithm for inequality constraints. This algorithm, which is due to Fourier [Sch86], is based on the following observation. Any non-strict inequality involving a variable x can be written in the form $x \leq r_u$ or $x \geq r_l$ i.e., it gives an upper or a lower bound on x. Thus if we are given two inequalities, one of the form $x \leq r_u$ and the other of the form $x \geq r_l$, we can eliminate x and obtain the inequality $r_l \leq r_u$. Obviously, $r_l \leq r_u$ is a logical consequence of the given inequalities. In addition, any solution of $r_l \leq r_u$ can be extended to a solution of the given inequalities (simply by choosing for x any value between the values of r_l and r_u). Following this observation, Fourier's elimination algorithm forms all pairs $x \leq r_u$ and $x \geq r_l$, eliminates x and returns the resulting constraints.

The same goal could have been achieved using the *adaptive consistency* techniques discussed by [DP88] in the context of finding backtrack-free solutions to binary constraint satisfaction problems. If we want to eliminate variables $x_1, \ldots, x_m$ from a set of inequalities C represented by a binary constraint network N then we could run the algorithm Adaptive-consistency of [DP88] on N with arguments $x_1, \ldots, x_m$. In the resulting network the set of constraints between variables $x_{m+1}, \ldots, x_n$ is equivalent to the set of constraints generated by Fourier's algorithm.

Fourier's algorithm can be extended to cope with disjunctions of inequations. The extension is based on the following observation. Consider the constraints $x \leq r_u, x \geq r_l$ and $x \neq r \vee \phi$ where ϕ is a disjunction of inequations or *false*. Without loss of generality we can assume that x does not appear in ϕ.[7] Let us eliminate x from the first two constraints to obtain $r_l \leq r_u$. Now consider a solution v of $r_l \leq r_u$. If v satisfies ϕ then it can be extended to a solution of the original constraints. If v does not satisfy ϕ then it can be extended to a solution of the original constraints if and only if r, r_l and r_u do not share the same value. As a result, the constraint $\neg\phi \supset r \neq r_u \vee r \neq r_l$ or equivalently $\phi \vee r \neq r_u \vee r \neq r_l$ must be included in the resulting set of constraints not involving x. Note that this constraint is also a consequence of the original constraints. Thus, if the variable to be eliminated is x, our algorithm reduces to the following statement: For all triples $x \leq r_u, x \geq r_l$ and $x \neq r \vee \phi$ of input constraints, add $r_l \leq r_u$ and $\phi \vee r \neq r_u \vee r \neq r_l$ in the output constraints. Thus as in lemma 3.1, disjunctions of inequations can be processed independently of one another when variable elimination is performed.

Example 4.1 Let us assume we want to eliminate variable x_1 from the set of constraints $\{x_3 \leq x_1,\ x_5 \leq x_1,\ x_1 \leq x_2,\ x_5 \neq x_1,\ x_4 \neq x_1\}$. Eliminating x_1 gives the inequalities $\{x_3 \leq x_2,\ x_5 \leq x_2\}$ and the disjunctions of inequations $\{x_5 \neq x_3 \vee x_5 \neq x_2,\ x_5 \neq x_5 \vee x_5 \neq x_2,\ x_4 \neq x_3 \vee x_4 \neq x_2,\ x_4 \neq x_5 \vee x_4 \neq x_2\}$. The latter set is equivalent to $\{x_5 \neq x_2,\ x_4 \neq x_3 \vee x_4 \neq x_2\}$.

Figure 3 gives our variable elimination algorithm. For simplicity, the algorithm assumes that one-variable inequalities are given as two-variable inequalities (with the introduction of an auxiliary variable x_0 which is understood to be equal to zero). With very few changes the same algorithm can be applied to the generalized linear constraints of [LM89]. The following proposition shows that the algorithm is correct. Its proof is an easy generalization of the previous discussion and can be found in [Kou92a].

Theorem 4.1 *Let C be a set of constraints in variables $x_1, x_2, \ldots, x_n$ and C' be the set of constraints produced by algorithm Variable_Elimination when x_1 is eliminated. If $(x_1^0, x_2^0, \ldots, x_n^0)$ is a solution of C then $(x_2^0, x_3^0, \ldots, x_n^0)$ is a solution of C'. Conversely, every solution $(x_2^0, x_3^0, \ldots, x_n^0)$ of C' can be extended to a solution $(x_1^0, x_2^0, \ldots, x_n^0)$ of C.*

[7] The constraint $x \neq r \vee \phi$ is equivalent to $\neg(x = r \wedge \neg\phi)$ where $\neg\phi$ is a conjunction of equalities. If we substitute r for x in $\neg\phi$, we obtain an equivalent constraint.

Algorithm Variable_Elimination

Inputs: A set of constraints $C = C_e \cup C_i \cup C_n$, in variables from the set X, in first canonical form. A set of variables E to be eliminated from C.

Output: A new set of constraints in variables from the set $E \setminus X$.

Method:
Initialize C'_e, C'_i and C'_n to $\emptyset$.
For each variable $x \in X$ do
  If x is the bound variable of equation $e_i \in C_e$ then
    Assign $(C_e - \{e_i\})$ to C_e, C'_i to C_i, C'_n to C_n
    Assign $\emptyset$ to C'_e, C'_i and C'_n
  ElseIf x is a parameter in equation $c_i : y_i = x + r$ of C_e then
    Rewrite c_i as $x = y_i - r$
    For each constraint c in $C_e - \{c_i\}$ (resp. C_i, C_n) do
      Substitute $y_i - r$ for x in c
      Add the resulting constraint to C'_e (resp. C'_i, C'_n)
    EndFor
    Assign C'_e to C_e, C'_i to C_i, C'_n to C_n
    Assign $\emptyset$ to C'_e, C'_i and C'_n
  Else (the variable x appears only in $C_i \cup C_n$)
    If there is at least one constraint $x - x_j \leq r_j$ in C_i then
      If there is at least one constraint $x_k - x \leq r_k$ in C_i then
        For each constraint $x_k - x \leq r_k$ in C_i do
          For each constraint $x - x_j \leq r_j$ in C_i $(j \neq k)$ do
            Add constraint $x_k - x_j \leq r_k + r_j$ in C'_i
            For each constraint $c \in C_n$ which contains x do
              Let c be $x - x_s \neq r_s \vee \phi$ such that ϕ does not involve x
              Simplify $x_k - x_s \neq r_k + r_s \vee x_j - x_s \neq r_s - r_j \vee \phi$
              and add it to C'_n.
            EndFor
          EndFor
        EndFor
        Add all constraints of C_i which do not contain x to C'_i
        Add all constraints of C_n which do not contain x to C'_n
      Else
        Add all constraints of C_i which do not contain x to C'_i
        Add all constraints of C_n which do not contain x to C'_n
      EndIf
    Else
      Add all constraints of C_i which do not contain x to C'_i
      Add all constraints of C_n which do not contain x to C'_n
    EndIf
    Assign C'_e to C_e, C'_i to C_i, C'_n to C_n
    Assign $\emptyset$ to C'_e, C'_i and C'_n
  EndIf
EndFor
Output $C_e \cup C_i \cup C_n$

Figure 3: An algorithm for variable elimination

Although the input to *Variable_Elimination* is in first canonical form, it is possible that its output is not. This can happen only if an eliminated variable is a parameter which appears only in $C_i \cup C_n$. When this variable is eliminated, C_i' does not contain implicit equalities (see [LHM89] for a proof of this) but may contain redundant constraints. For example, assume C_i is $\{x_2 \leq x_1,\ x_1 \leq x_3,\ x_2 \leq x_5,\ x_5 \leq x_3\}$ and the eliminated variable is x_1. Then C_i' is $\{x_2 \leq x_3,\ x_2 \leq x_5,\ x_5 \leq x_3\}$ and obviously $x_2 \leq x_3$ is redundant. There are also cases when the constraints C_n' are not precise or irredundant. For example, assume that $C_i \cup C_n$ is $\{x_3 - x_2 \leq 2,\ x_2 - x_1 \leq 3,\ x_7 \geq 5,\ x_3 \leq 0,\ x_2 \neq x_7\}$. After x_2 is eliminated, we are left with $\{x_3 - x_1 \leq 5,\ x_7 \geq 5,\ x_3 \leq 0,\ x_3 - x_7 \neq 2 \vee x_1 - x_7 \neq -3\}$. It is easy to see now that $x_3 - x_7 \neq 2$ is implied by the inequality constraints. Consequently, if the output of *Variable_Elimination* must be returned in first canonical form, redundancy must be eliminated from C_i', each $c \in C_n'$ must be made precise and finally redundancy must be eliminated from C_n' (in this order!).

Fortunately, in the remaining cases the output of *Variable_Elimination* is in first canonical form as it is demonstrated by the following theorem (its proof can be found in [Kou92a]).

Theorem 4.2 *Let the inputs to algorithm Variable_Elimination be $C_e \cup C_i \cup C_n$ and X. If all variables of X are bound or parameter variables of the input set C_e then the output sets C_i and C_n are irredundant. Moreover, the former set does not contain implicit equalities and the latter consists only of precise constraints.*

In what follows, we will assume that when a variable is eliminated by *Variable_Elimination*, the resulting constraint set $C_e' \cup C_i' \cup C_n'$ is transformed into first canonical form before proceeding with the elimination of the rest of the variables.[8] Let us now investigate the complexity of *Variable_Elimination*.

Lemma 4.1 *Let $C = C_i \cup C_n$ be a set of temporal constraints. If m variables appearing in C_i are eliminated using algorithm Variable_Elimination and the resulting set is $C' = C_i' \cup C_n'$ then $|C_i'| \leq 2(v(C_i) - m)^2$ and $|C_n'| \leq |C_n|\,(v(C_i) - 1)^{2m}$.*

Theorem 4.3 *The algorithm Variable_Elimination takes time polynonial in the size of the input constraint set and exponential in the number of eliminated variables.*

The following theorem shows that variable elimination is intractable. The source of this intractability is the fact that the size of the output constraint set can be exponential in the number of eliminated variables. The following section examines a tractable case.

Theorem 4.4 *For every m, k such that $k \geq 4m$ we can find a set of temporal constraints C of size $\|C\| \leq k$ and m variables of C such that when these variables are eliminated the resulting constraint set cannot be represented by fewer than 2^m disjunctions of inequations and $2m$ inequalities.*

Let us now consider variable elimination when the input constraint set is in second canonical form. In this case, elimination of one (or many) variables is trivial if these variables appear *only* in the inequality constraints. This is a consequence of the fact that the minimal network for a set of inequality constraints is decomposable (this concept will be explained fully in the following section) [DMP91]. In the remaining cases, the situation is as for the first canonical form.

5 VARIABLE ELIMINATION FOR SIMPLE TEMPORAL CONSTRAINTS

In this section, we show that for simple temporal constraints variable elimination is tractable. Our proof is indirect and is based on the fact that a set of simple temporal constraints can be transformed into an equivalent *decomposable* constraint set in time polynonial in its size. This indirect proof is interesting in its own right since it answers an open problem of [vB90a].

A set of constraints C is called *decomposable* if and only if it is λ-decomposable for every λ, $1 \leq \lambda \leq v(C)$ [DMP91]. A set of constraints is λ-decomposable if and only if every valuation $u = \{x_1 \leftarrow x_1^0, \ldots, x_{\lambda-1} \leftarrow x_{\lambda-1}^0\}$ of $\lambda - 1$ variables which satisfies the constraints applicable to $x_1, \ldots, x_{\lambda-1}$ (namely, those involving only these variables) can be extended to a valuation $u' = \{x_1 \leftarrow x_1^0, \ldots, x_{\lambda-1} \leftarrow x_{\lambda-1}^0, x_\lambda \leftarrow x_\lambda^0\}$ (for any variable x_λ) such that u' satisfies the constraints applicable to $x_1, \ldots, x_\lambda$.

The algorithm *Decompose* in figure 4 transforms a set of simple temporal constraints into an equivalent decomposable constraint set (which is not necessarily simple). The algorithm proceeds in three steps. At first, all equalities and inequalities which are consequences of C_e and C_i are computed by *Deductive_Closure*. This algorithm is not presented. For the case of C_e it is trivial; for the case of C_i the path consistency algorithm of [vB90a] (for the point algebra without $\neq$) can be used to achieve the same effect. The set returned by *Deductive_Closure* is decomposable [vB90a]. In its second step, *Decompose* computes a decomposable constraint set $C_i' \cup C_n'$ equivalent to $C_i \cup C_n$. This is performed in a controlled manner to avoid generating redundant constraints (see

[8] In practice, it must be more efficient to simply remove redundancy every time a variable is eliminated and postpone the transformation of inequation constraints to precise ones until the end of the algorithm.

Algorithm Decompose

Input: A set of simple temporal constraints $C = C_e \cup C_i \cup C_n$ in first canonical form.

Output: A decomposable set of constraints equivalent to C.

Method:
$C'_e := Deductive\_Closure(C_e)$
$C'_i := Deductive\_Closure(C_i)$

$C_n^1 := \emptyset$; $C_n^2 := \emptyset$
For every c: $x_i \neq x_k$ in C_n do
  For every pair of constraints $x_l \leq x_i$ and $x_i \leq x_m$ in C'_i do
    Let c' be $x_k \neq x_l \vee x_m \neq x_l$ simplified
    Mark x_k in c'
    Add c' to C_n^1
  EndFor
  For every pair of constraints $x_l \leq x_k$ and $x_k \leq x_m$ in C'_i do
    Let c' be $x_i \neq x_l \vee x_m \neq x_l$ simplified
    Add c' to C_n^2
  EndFor
EndFor

$C_n^3 := \emptyset$
For every c: $x_k \neq x_l \vee x_m \neq x_l$ in C_n^1 such that x_k is marked do
  For every pair of constraints $x_t \leq x_k$ and $x_k \leq x_s$ in C'_i do
    Let c' be $x_t \neq x_l \vee x_s \neq x_l \vee x_m \neq x_l$ simplified. Add c' to C_n^3
  EndFor
EndFor
$C'_n := C_n^1 \cup C_n^2 \cup C_n^3$

$C_{new} := Implied\_Constraints(C'_e, C'_i \cup C'_n)$

Output $C'_e \cup C_{new}$

Figure 4: The algorithm *Decompose*

the following lemmas). Finally, *Decompose* takes into account the equalities in C'_e and generates the constraints implied by the ones in $C'_i \cup C'_n$. This is done by function *Implied_Constraints* which is not presented since it is straightforward: C_{new} will contain one constraint for each constraint $c \in C'_i \cup C'_n$ in parameters $x_1, \ldots, x_l (l \leq 5)$ and each combination of (bound or parameter) variables $z_1, \ldots, z_l$ such that $z_i = x_i \in C'_e$, $i = 1, \ldots, l$.

To show that the algorithm is correct, we need the following lemmas. Appendix A contains the proof for lemma 5.2. The proofs for the rest of the lemmas are similar.

Lemma 5.1 *Let $c : z \neq y \vee \phi$ be a constraint in C_n^1 such that z has not been marked and ϕ does not contain z. If $x_a \leq z$ and $z \leq x_b$ are constraints of C'_i then there exists a constraint $c' \in C_n^1$ such that $c' \models x_a \neq y \vee x_b \neq y \vee \phi$.*

Lemma 5.2 *Let $c : z \neq y \vee \phi$ be a constraint in C_n^2 such that ϕ does not contain z. If $x_a \leq z$ and $z \leq x_b$ are constraints of C'_i then there exists a constraint $c' \in C_n^2 \cup C_n^3$ such that $c' \models x_a \neq y \vee x_b \neq y \vee \phi$.*

Lemma 5.3 *Let $c : z \neq y \vee \phi$ be a constraint in C_n^3 such that ϕ does not contain z. If $x_a \leq z$ and $z \leq x_b$ are constraints of C'_i then there exists a constraint $c' \in C_n^3$ such that $c' \models x_a \neq y \vee x_b \neq y \vee \phi$.*

The following lemma is used in computing the complexity of algorithm *Decompose*.

Lemma 5.4 *If $C = C_e \cup C_i \cup C_n$ is a set of simple temporal constraints in first canonical form then Decompose returns a constraint set $C'_e \cup C_{in}$ such that*

$$|C'_e| \leq \frac{1}{2}(v(C_e)^2 - v(C_e)) \text{ and } |C_{in}| \leq |C_e|^5 \, v(C_i)^4 \, |C_n|.$$

We will now state the main result of this section.

Theorem 5.1 *The algorithm Decompose takes time $O(|C_e|^5 \, v(C_i)^4 \, |C_n|)$ and computes a decomposable constraint set equivalent to the input one.*

In fact, the previous theorem overestimates the running time of *Decompose*. As can be seen from the proof of lemma 5.4, a more precise upper bound is $O(k^5 v(C_i)^4 \, |C_n|)$ where k is the maximum number of equations of C_e which involve the same parameter variable.

Variable elimination is a trivial operation for decomposable sets of constraints. If C is a decomposable set of constraints then variables $x_1, \ldots, x_m$ can be eliminated from C by simply deleting all constraints of C which involve these variables. However, lemma 5.4 implies that even in the case of simple temporal constraints, a decomposable constraint set can contain a rather big number of disjunctions of inequations. Therefore this representation will probably be avoided in practise.

The above observation has also the following consequence. If a variable is eliminated from a set of simple temporal constraints, using *Variable_Elimination*, the first canonical form of the resulting set has size smaller than the size of the set obtained by first invoking *Decompose* and then eliminating this variable. The following corollary is now obvious.

Corollary 5.1 *For simple temporal constraints, algorithm Variable_Elimination takes time polynomial in the size and the number of variables of the input constraint set (thus, it also takes time polynomial in the number of eliminated variables).*

6 CONCLUSIONS

This paper discusses consistency checking, canonical forms and variable elimination for temporal constraints. Our results extend previous work on temporal constraints by considering disjunctions of inequations. This extension has been motivated by the fact that in certain temporal formalisms (e.g., [DM87, Kou92b]) allowing inequations and disallowing disjunctions of inequations compromises the ability to answer certain queries. Although the emphasis of this paper is rather theoretical, we plan to investigate the performance of our algorithms in a real temporal reasoning system based on the temporal database model of [Kou92b]. Among other things, this will allow us to determine cases in which the first canonical form is more appropriate than the second one and vice versa.

This work can be extended in several ways. We are currently considering other special cases of temporal constraints with the intention to determine the exact complexity of variable elimination for these cases. We are also investigating the behaviour of our algorithms in cases where constraints are added to the system incrementally.

It would be interesting to study variable elimination in temporal models involving both intervals and points. Currently, we can solve this problem by translating interval constraints to endpoint constraints and then performing variable elimination. The challenge would be to devise algorithms which avoid this translation.

Finally, our framework must be extended to deal with other kinds of non-binary temporal constraints. For example, "the duration of interval I exceeds the duration of interval J" which is a constraint on four points (the endpoints of I and J).

Acknowledgements

Most of this work was performed while the author was on leave from the Dept. of Computer Science, University of Toronto and was visiting the Dept. of Electrical and Computer Engineering, National Technical University of Athens and the Institute of Computer Science, Foundation of Research and Technology - Hellas. Financial support was gratefully received from the Institute of Computer Science, Foundation of Research and Technology - Hellas and the Dept. of Computer Science, University of Toronto. The following people provided important help during different stages of this work: Foto Afrati, Costas Courcoubetis, Jean-Louis Lassez, Itay Meiri, Dimitris Plexousakis, Timos Sellis, Thodoros Topaloglou, Peter van Beek and Yannis Vassiliou. In addition, one of the referees provided comments that allowed us to improve our presentation.

References

[All83] J.F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[DM87] T. Dean and D.V. McDermott. Temporal Data Base Management. *Artificial Intelligence*, 32:1–55, 1987.

[DMP91] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49(1-3):61–95, 1991. Special Volume on Knowledge Representation.

[DP88] Rina Dechter and Judea Pearl. Network-Based Heuristics for Constraint Satisfaction Problems. *Artificial Intelligence*, 34(1):1–38, 1988.

[End72] H.B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[KL91] H. Kautz and P. Ladkin. Integrating Metric and Qualitative Temporal Reasoning. In *Proceedings of AAAI-91*, pages 241–246, 1991.

[Kou92a] Manolis Koubarakis. Dense Time and Temporal Constraints with ≠. Long version of this paper. Forthcoming., 1992.

[Kou92b] Manolis Koubarakis. Representation and Querying in Temporal Databases: the Power of Temporal Constraints. Forthcoming., 1992.

[LHM89] Jean-Louis Lassez, Tien Huynh, and Ken McAloon. Simplification and Elimination of Redundant Linear Arithmetic Constraints. In *Proceedings of NACLP-89*, pages 37–51. MIT Press, 1989.

[LM88] Peter Ladkin and Roger Maddux. On Binary Constraint Networks. Technical Report KES.U.88.8, Kestrel Institute Technical Report, 1988.

[LM89] Jean-Louis Lassez and Ken McAloon. A Canonical Form for Generalized Linear Costraints. Technical Report RC15004 (#67009), IBM Research Division, T.J. Watson Research Center, 1989.

[Mac77] A.K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[MBJK90] John Mylopoulos, Alex Borgida, Matthias Jarke, and Manolis Koubarakis. Telos: A Language for Representing Knowledge About Information Systems. *ACM Transactions on Information Systems*, 8(4):325–362, October 1990.

[Mei91] I. Meiri. Combining Qualitative and Quantitative Constraints in Temporal Reasoning. Technical Report R-160, Cognitive

Systems Laboratory, University of California, Los Angeles, 1991.

[Mon74] U. Montanari. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Sciences*, 7:95–132, 1974.

[Rei80] Raymond Reiter. Equality and Domain Closure in First Order Databases. *Journal of the ACM*, 27(2):235–249, 1980.

[Sch86] A. Schrijver, editor. *Theory of Integer and Linear Programming*. Wiley, 1986.

[SG88] Yoav Shoham and Nita Goyal. Temporal Reasoning in Artificial Intelligence. In Strobe, Howard and AAAI, editor, *Exploring Artificial Intelligence*, pages 419–439. Morgan Kaufmann, 1988.

[Sno90] R. Snodgrass. Temporal Databases: Status and Research Directions. *Sigmod Record*, 19(4):83–89, 1990.

[vB90a] Peter van Beek. Exact and Approximate Reasoning About Qualitative Temporal Relations. Technical Report TR 90-29, Department of Computing Science, University of Alberta, August 1990.

[vB90b] Peter van Beek. Reasoning About Qualitative Temporal Information. In *Proceedings of AAAI-90*, pages 728–734, 1990.

[VKvB89] Marc Vilain, Henry Kautz, and Peter van Beek. Constraint Propagation Algorithms for Temporal Reasoning: a Revised Report. In D.S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann, 1989.

A PROOFS

Lemma 4.1: When m variables are eliminated we are left with $v(C) - m$ variables. Therefore lemma 3.2 implies $|C_i| \leq 2(v(C_i) - m)^2$.

For the second inequality, let C_i^l (resp. C_n^l) be the set of inequalities (resp. disjunctions of inequations) produced by *Variable_Elimination* when the l-th variable is eliminated ($1 \leq l \leq m$). When the l-th variable x is eliminated there are at most $2\binom{v(C_i)-l}{2}$ pairs of constraints $x_k - x \leq r_k$, $x - x_j \leq r_j$ in C_i^l. Therefore $|C_n^1| \leq |C_n|(v(C_i)-1)^2$, $|C_n^2| \leq |C_n^1|(v(C_i)-2)^2 \leq |C_n|(v(C_i)-1)^2(v(C_i)-2)^2$ and so on. Finally, $|C_n^m| \leq |C_n| \prod_{l=1}^{m}(v(C_i)-l)^2 \leq |C_n|(v(C_i)-1))^{2m}$.

Theorem 4.3: Assume C and $C' = C'_e \cup C'_i \cup C'_n$ are the input and output sets and m is the number of eliminated variables. Notice that every step of *Variable_Elimination* takes time polynomial in the size of the constraint sets generated in this particular step.

Lemmas 3.2 and 4.1 give $|C'_e| \leq |C_e|$, $|C'_i| \leq 8v(C_i)^2 \leq 2\|C\|^2$ and $|C'_n| \leq \|C_n\|(2\|C_i\|)^{2m} \leq 2^{2m}\|C\|^{2m+1}$. The theorem now follows because $\|C'_e\| = |C'_e|$, $\|C'_i\| = |C'_i|$ and $\|C'_n\| \leq |C'_n|\, v(C'_n)$.

Theorem 4.4: Given m, k such that $k \geq 4m$, the set C in variables $L_1, \ldots, L_m$, $U_1^1, \ldots, U_m^1$, $U_1^2, \ldots, U_m^2$, $X_1, \ldots, X_m$, $Y_1, \ldots, Y_m$ is $C = \bigcup_{1 \leq i \leq m} C_i \cup C_n$ where $C_i = \{L_i \leq X_i,\ X_i \leq U_i^1,\ X_i \leq U_i^2\}$ and $C_n = \{\bigvee_{i=1}^m Y_i \neq X_i\}$. Let us now eliminate variables $X_1, \ldots, X_m$ to arrive at $C' = \bigcup_{1 \leq i \leq m} C'_i \cup C'_n$ where $C'_i = \{L_i \leq U_i^1,\ L_i \leq U_i^2\}$ and $C_n = \{\bigvee_{i=1}^m Y_i \neq L_i \vee Y_i \neq U_i^{j_i} :\ j_i = 1, 2\}$. The first canonical form of C' is $C'' = \bigcup_{1 \leq i \leq m} C'_i \cup C''_n$ where

$$C''_n = \{\bigvee_{i=1}^{m} Y_i \neq L_i \vee U_i^{j_i} \neq L_i :\ j_i = 1, 2\}$$

if the order of the variables is the one given above. Obviously C' (and C'') contains 2^m disjunctions of inequations and $2m$ inequalities. Now assume that there exists a set of temporal constraints A such that $A \neq C'$, $Sol(C') = Sol(A)$ and $|A| < 2^m + 2m$ (proof by contradiction). If A' is the first canonical form of A then $|A'| \leq |A|$. But equivalent sets of constraints have identical canonical forms therefore $A' = C''$ thus $|A| \geq 2^m + 2m$ which contradicts the assumption $|A| < 2^m + 2m$.

Lemma 5.2: Every constraint in C_n^2 is of the form $x_i \neq x_l \vee x_m \neq x_l$ and has been generated by considering inequalities $x_l \leq x_k$ and $x_k \leq x_m$ of C'_i and the inequation $x_i \neq x_k$ of C_n (for some fixed i, k, l, m). We now have the following three cases to consider:

1. $z = x_l$, $y = x_i$.
   Let us consider any pair of inequalities $x_a \leq x_l$, $x_l \leq x_b$ from C'_i and the disjunction of inequations $x_l \neq x_i \vee x_m \neq x_i$. These constraints imply $c'' :\ x_a \neq x_i \vee x_b \neq x_i \vee x_m \neq x_i$. But notice that $x_a \leq x_k \in C'_i$. Thus *Decompose* considers $x_a \leq x_k$, $x_k \leq x_m$ and $x_k \neq x_i$ and adds the constraint $c' :\ x_a \neq x_i \vee x_m \neq x_i$ to C_n^2. Obviously, $c' \models c''$.
2. $z = x_i$, $y = x_l$.
   Let us consider any pair of inequalities $x_a \leq x_i$, $x_i \leq x_b$ from C_i and the disjunction $x_i \neq x_l \vee x_m \neq x_l$. These constraints imply $c'' :\ x_a \neq x_l \vee x_b \neq x_l \vee x_m \neq x_l$. But if we consider $x_a \leq x_i$, $x_i \leq x_b$ and $x_i \neq x_k$, we can see that $x_k \neq x_a \vee x_b \neq x_a \in C_n^1$ and x_k is marked. Therefore *Decompose* considers

$x_l \leq x_k$, $x_k \leq x_m$ and $x_k \neq x_a \vee x_b \neq x_a$ and adds $c' : x_a \neq x_l \vee x_m \neq x_a \vee x_b \neq x_a$ to C_n^3. Obviously, $c' \models c''$.

3. $z = x_m$, $y = x_l$.
Let us consider any pair of inequalities $x_a \leq x_m$, $x_m \leq x_b$ from C_i and the disjunction $x_m \neq x_l \vee x_i \neq x_l$. These constraints imply $c'' : x_a \neq x_l \vee x_b \neq x_l \vee x_i \neq x_l$. But notice that $x_k \leq x_b \in C_i'$. Therefore *Decompose* considers $x_l \leq x_k$, $x_k \leq x_b$ and $x_k \neq x_i$ and adds $c' : x_l \neq x_i \vee x_b \neq x_i$ to C_n^2. Obviously, $c' \models c''$.

Lemma 5.4: There can be at most $\binom{v(C_e)}{2}$ constraints in C_e'. Thus $|C_e'| \leq \frac{1}{2}(v(C_e)^2 - v(C_e))$. Similarly, $|C_i'| \leq \frac{1}{2}(v(C_i)^2 - v(C_i))$.

For every variable x there are at most $\frac{1}{4}(v(C_i) - 1)^2$ pairs of constraints $y \leq x$, $x \leq z$. Therefore $|C_n^1| \leq \frac{1}{4}(v(C_i) - 1)^2 |C_n|$, $|C_n^2| \leq \frac{1}{4}(v(C_i) - 1)^2 |C_n|$ and $|C_n^3| \leq \frac{1}{16}(v(C_i) - 1)^4 |C_n|$. Finally,

$$|C_n'| \leq \frac{1}{16}(v(C_i) - 1)^4 |C_n|.$$

Let us now turn to the function *Implied_Constraints*. Let k denote the maximum number of equations of C_e which involve the same parameter variable. Obviously $k \leq |C_e|$. Notice now that every constraint in C_n' involves at most 5 variables. From each such constraint, at most $(k+1)^5$ constraints can be generated in C_{new}. Similarly, from each constraint in C_i' (which involves at most 2 variables), at most $(k+1)^2$ constraints can be generated in C_{new}.

Therefore $|C_{in}| \leq |C_e|^2 |C_i'| + |C_e|^5 |C_n'| \leq |C_e|^5 v(C_i)^4 |C_n|$.

Theorem 5.1: For every constraint set C and set of variables $X = \{x_1, \ldots, x_n\}$, the set of constraints in C which involve only variables from X will be denoted by $C(X)$ or $C(x_1, \ldots, x_n)$.

Initially we will show that $C_i' \cup C_n'$ is decomposable. $C_i' \cup C_n'$ is 1-decomposable since it is consistent. We will now prove that it is decomposable for each λ such that $2 \leq \lambda \leq v(C_i' \cup C_n')$. Let us assume that there is a λ, $2 \leq \lambda \leq v(C_i' \cup C_n')$ such that $C_i' \cup C_n'$ is not λ-decomposable (proof by contradiction). In other words, there exists a valuation $u = \{x_1 \leftarrow x_1^0, \ldots, x_{\lambda-1} \leftarrow x_{\lambda-1}^0\}$ and variable x_λ such that u satisfies $C_i' \cup C_n'(x_1, \ldots, x_{\lambda-1})$ but there is no value x_λ^0 such that $u' = u \cup \{x_\lambda \leftarrow x_\lambda^0\}$ satisfies $C_i' \cup C_n'(x_1, \ldots, x_{\lambda-1}, x_\lambda)$.

Let us now observe that *Deductive_Closure* returns a decomposable constraint set C_i'. Therefore $C_i' \cup C_n'(x_1, \ldots, x_{\lambda-1}, x_\lambda)$ must contain at least one constraint from C_n. Let us apply valuation u to $C_i' \cup C_n'(x_1, \ldots, x_{\lambda-1}, x_\lambda)$ to obtain $C_i' \cup C_n'(x_1^0, \ldots, x_{\lambda-1}^0, x_\lambda)$. But now lemma 3.1 implies that there exists a disjunction of inequations $c_0 \in C_n'(x_1^0, \ldots, x_{\lambda-1}^0, x_\lambda)$ such that $C_i'(x_1^0, \ldots, x_{\lambda-1}^0, x_\lambda) \cup \{c_0\}$ is inconsistent. Let c_0 be $x_\lambda \neq x_\rho^0$ and $C_i'(x_1^0, \ldots, x_{\lambda-1}^0, x_\lambda)$ be $x_\mu^0 \leq x_\lambda$, $x_\lambda \leq x_\nu^0$ where $1 \leq \rho, \mu, \nu \leq \lambda - 1$.

Since we have an inconsistency $x_\mu^0 = x_\nu^0 = x_\rho^0$ must hold. Let us assume that $c : x_\lambda \neq x_\rho \vee \phi$ is the constraint of $C_n'(x_1, \ldots, x_\lambda)$ which results in c_0 after u is applied (ϕ does not contain x_λ).Now we have to consider the following cases:

1. $c \in C_n$ or $c \in C_n^1$ and x_λ has been marked. Then *Decompose* considers $x_\mu \leq x_\lambda, x_\lambda \leq x_\nu$ and $x_l \neq x_\rho \vee \phi$ and generates $c' : x_\mu \neq x_\rho \vee x_\nu \neq x_\rho \vee \phi$ in C_n'. Let us now note that u satisfies c' but falsifies ϕ. Thus $x_\mu^0 \neq x_\rho^0 \vee x_\nu^0 \neq x_\rho^0$ must hold and therefore we have a contradiction with $x_\mu^0 = x_\nu^0 = x_\rho^0$. Thus the original assumption is false.
2. $c \in C_n^1$ and x_λ is not marked. The proof for this case is similar to the proof for the following case. We only use lemma 5.1 instead of lemma 5.2.
3. $c \in C_n^2$ and c is $x_\lambda \neq x_\rho \vee x_\sigma \neq x_\rho$ where $1 \leq \sigma \leq \lambda - 1$. In this case, lemma 5.2 tells us that there is a constraint $c' \in C_n^2 \cup C_n^3$ such that $c' \models x_\mu \neq x_\rho \vee x_\nu \neq x_\rho \vee x_\sigma \neq x_\rho$. Obviously c' involves only variables from $\{x_1, \ldots, x_{\lambda-1}\}$ and it is satisfied by u. Thus u must also satisfy c'. But observe that the form of c_0 implies $x_\sigma^0 = x_\rho^0$. Therefore $x_\mu^0 \neq x_\rho^0 \vee x_\nu^0 \neq x_\rho^0$ which contradicts our assumption.
4. $c \in C_n^3$.
The proof is similar to the one given for the previous case. We now have to use lemma 5.3.

It is not difficult to see now that $C_e' \cup C_{in}$ is decomposable.

Finally, let us calculate the time complexity of *Decompose*. The calls to *Deductive_Closure* take $O(v(C_i)^3)$ time. The rest of the computation depends only on the sizes of the constraint sets generated. Therefore lemma 5.4 implies that the algorithm takes $O(|C_e|^5 v(C_i)^4 |C_n|)$ time.

Managing Disjunction for Practical Temporal Reasoning

Robert Schrag* **Mark Boddy** **Jim Carciofini**
{boddy | carciofi | schrag}@src.honeywell.com
Honeywell Systems & Research Center, MN65-2100
3660 Technology Drive
Minneapolis, MN 55418 USA

Abstract

Ambiguous conclusions are inescapable in temporal reasoning. Lack of precise information about what events happen when results in uncertainty regarding the events' effects. Incomplete information and nonmonotonic inference result in situations where there is more than one set of possible conclusions, even when there is no temporal uncertainty at all. In an implemented system, this ambiguity is a computational problem as well as a semantic one. We discuss some of the sources of this ambiguity, which we treat as explicit *disjunction*, in the sense that ambiguous information can be interpreted as defining a set of possible inferences. Three ways of handling this disjunction are to represent it explicitly, to remove it by limiting the expressive power of the system, or to approximate a set of disjuncts using a weaker form of representation. We have employed primarily the latter two of these approaches to implement an expressive and efficient temporal reasoning engine that performs sound inference in accordance with a well-defined formal semantics.

1 INTRODUCTION

Lack of precise information about what facts are true when or what events happen when leads to ambiguity about what is true at a given point in time. This can be either direct (*e.g.*, the library closed between 3:00 and 4:00, and I got there at 3:30) or the result of projection or other inference (*e.g.*, if my cab ride takes too long and I miss my plane, I will not be in Washington tomorrow). Nonmonotonic inference (*e.g.*, the persistence assumption or qualified causal projection) leads to situations where there is more than one set of possible conclusions, even when there is no temporal uncertainty at all [Hanks and McDermott, 1986]. In a formal system, this ambiguity is noted and dealt with in some way, either by changing the semantics to exclude it (*e.g.*, by assigning a preference relation to the possible models of a given theory [Shoham, 1988]), or by simply acknowledging it (*i.e.*, couching conclusions in terms of the set of possible models [Kautz, 1986]). In an implemented system, this ambiguity is a computational problem as well as a semantic one.

In this paper, we discuss some of the sources of this ambiguity, which we treat as explicit *disjunction*, in the sense that ambiguous information can be interpreted as defining a set of possible conclusions. We describe how these sources of disjunction are dealt with in our current implementation of Dean's *Time Map Manager* [Dean and McDermott, 1987, Dean, 1986]. Briefly, we take one of three approaches.

- We represent the disjunction explicitly.
- We remove the disjunction by limiting the expressive power of the system.
- We approximate the disjunction by a weaker form of representation that subsumes the disjunction.

The first of these approaches must be applied carefully, because of the possibility of combinatorial explosion in the set of disjunctions that must be represented. The only representation of explicit disjunction we are considering in TMM is in a context-switching mechanism that would put responsibility for managing disjunctive contexts largely in the user's hands; we briefly discuss this approach in Section 6.

Primarily we make use of the latter two approaches in the current TMM. We have found the use of weak representations to be of more general utility than was first realised: the first application of this approach was to model quantification over a set of total orders approximately [Dean and Boddy, 1988]. We recently have extended the same approach (in fact using the same representations) to represent disjunction arising from

*Author's current address: Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712 USA — schrag@cs.utexas.edu

aspects of the system semantics, specifically from the use of the persistence assumption; we describe these extensions in Section 5.

In the rest of this paper, we briefly discuss the ontology and semantics of TMM, provide some specific examples of the kinds of disjunction that arise, and demonstrate the application and limitations of this approach.

2 TMM

Dean's *Time Map Manager* (TMM) [Dean and McDermott, 1987, Dean, 1986] is an implemented temporal reasoning system, intended as a foundation for building planning and scheduling systems. TMM includes capabilities for reasoning about partially ordered events, persistence and clipping, and simple causal reasoning, all in the presence of incremental change in a dynamic environment. We present a simplified but accurate description of the system's ontology and semantics below.

2.1 ONTOLOGY

A *domain theory* $\mathcal{D}$ includes a *time map* and a *causal theory*.

The time map consists of a set of *time points* $\mathcal{T}$ and a set of formulas. Time map formulas include the following.

- ***Temporal relations* between time points, denoted by the binary infix predicates $<$, $\leq$, $=$, $\geq$, and $>$, and the predicate distance(t1, t2, *bounds*), where $t1, t2 \in \mathcal{T}$ and *bounds* = [r1 r2], where r1, r2 $\in \Re$ are the bounds of a convex, closed interval.[1] We represent temporal relations in our implementation as *constraints*.**
- ***Temporal propositions,* holds_o(t1, t2, P), where $t1, t2 \in \mathcal{T}$ and $P \in \mathcal{P}$, the set of *propositions*. The interval between t1 and t2 is called the *observation interval,* throughout which the proposition is believed to hold "by observation" and is not defeasible. We refer to t1 as the "begin" point of the temporal proposition; to t2 as the "end" point. Generically we refer to both as "endpoints." We represent temporal propositions in our implementation using *time tokens*.**

  In TMM all temporal propositions hold by observation for every point in the observation interval.[2]
- ***Persistence assumptions,* persists_b(t1, P) and persists_f(t2, P), where t1, t2, and P appear in some instance of the predicate holds_o as above, associating persistence with a specific temporal proposition. persists_b and persists_f refer to backward and forward persistence, respectively. We associate persistence assumptions with time tokens in our implementation.**

The causal theory includes *causal rules,* intended to encode the physics of a domain.

- ***Projection rules,* project(($\bigwedge \langle P_1, \ldots, P_k \rangle$), E, R). The propositions $P_1, \ldots, P_k$ are "antecedent" propositions; E is a "trigger" proposition; R is a forward-persistent "result" proposition. When the conjunction of antecedents is believed to hold throughout the trigger, the result also is believed to hold starting immediately after the trigger. We represent projection result propositions with time tokens on the time map.**
- ***Overlap chaining rules,* $(\bigwedge \langle P_1, \ldots, P_k \rangle) \Rightarrow_t R$. The propositions $P_1, \ldots, P_k$ are antecedents; the proposition R is a result. Whenever the conjunction of antecedents is believed to hold, the result also is believed to hold at the same time. This was described in [Dean, 1986]. (We have experimented with overlap chaining in earlier β-TMM prototypes, but we do not include it in the implementation described in [Schrag *et al.*, 1992].)**

2.2 TIME MAP DISPLAY

We display time maps in the examples to follow using a pictorial notation. We represent a time point t by an optionally labeled dot: $\underset{t}{\bullet}$. We represent the observation interval of a time token with proposition P by two time points, a connecting line, and optionally a label: ●———● P. When the observation interval has zero length it it will appear as a single point. We represent persistence assumptions by backward and forward pointing arrows on the time token's observation interval: ◄●———●►. All time points are drawn with respect to a fixed frame of reference where left-to-right positioning indicates relative distance. Uncertainty in a time point's position is represented by a dashed line: ●------●——●. In this case, the begin point could be anywhere along the dashed line.

2.3 SEMANTICS

TMM implements an epistemic semantics, in the sense that we may believe a proposition to hold at a point, or we may believe it not to hold at that point, or we may not believe either way. The failure of the excluded middle in this semantics is useful for representing problems where we have only partial information. Inference from causal rules results in the addition of new temporal propositions to the time map, representing belief in propositions holding for new intervals of time. Persistence is captured in a preference over models:

---

[1]Our implementation uses infinitesimals in a non-standard calculus that achieves the same effects as intervals open on either end when desired.

[2]In the terminology of Shoham [Shoham, 1987] our temporal propositions are called "liquid."

we prefer models in which the appropriate facts persist over those in which they don't.[3] When different, equally plausible models exist for a given theory (none of which is preferred over all the others) ambiguity results. Partial orders and persistence are the direct sources of ambiguity in our system.

Given a domain theory, we are interested in the following kinds of conclusions.[4]

- $\text{holds}_{\Box}$(t1, t2, P): P holds necessarily over the interval between points t1 and t2.
- $\text{holds}_{\Diamond}$(t1, t2, P): P holds possibly over the interval.
- Inferences about necessary temporal relations, *e.g.*, $t3 <_{\Box} t4$ (t3 is necessarily before t4).
- Inferences about possible temporal relations, *e.g.*, $t3 <_{\Diamond} t4$.
- Boolean combinations of these.

To answer these queries, we also need the following predicates.

- $\text{holds}_{\circ\Box}$(t1, t2, P): P holds necessarily by observation over the interval between t1 and t2.
- $\text{holds}_{\circ\Diamond}$(t1, t2, P): P holds possibly by observation over the interval.

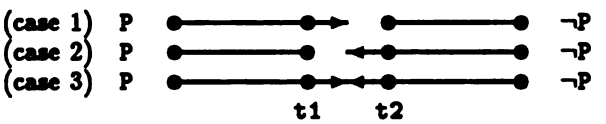
For each of the holds predicates (holds, $\text{holds}_{\circ}$, $\text{holds}_{\Box}$, $\text{holds}_{\Diamond}$, $\text{holds}_{\circ\Box}$, and $\text{holds}_{\circ\Diamond}$), we have the following equivalences.

$$\text{holds}(t1, t2, P) \equiv (t1 \leq t \leq t2 \Rightarrow \text{holds}(t, P))$$

$$\text{holds}(t, P) \equiv \text{holds}(t, t, P)$$

We define necessity and possibility using quantification over the *possible worlds*, or sets of conclusions, consistent with the user-supplied domain theory: a conclusion is true necessarily iff it is true in every possible world (universal quantification); it is true possibly iff it is true in some possible world (existential quantification). Within any given possible world, we may ask merely holds(t1, t2, P), $\text{holds}_{\circ}$(t3, t4, P), or $t5 < t6$, without quantification. A set of temporal relations that defines only a partial order on the set of time points represents the simplest form of ambiguity in the domain theory; we will discuss some cases of this in detail in Section 4. The nonmonotonic persistence assumption results in more complicated forms of ambiguity; we will discuss some cases of these in Section 5. Our definitions for $\text{holds}_{\Box}$ and $\text{holds}_{\Diamond}$ depend on temporal relations, temporal propositions, causal rules, and persistence. For temporal relations, the different possible worlds are the *total orders* consistent with a given partial order. All of the temporal propositions (observations) in the domain theory hold necessarily. The results of causal rules whose antecedents (and, for projection rules, the trigger event) hold necessarily also hold necessarily; the results of those whose antecedents hold possibly also hold possibly. We define the semantics for persistence below.

The ultimate extent of a temporal proposition's persistence in either direction, backward or forward, is defined only in terms of when it is defeated by another temporal proposition. In this paper the only form of defeat we discuss is *clipping*, in which a temporal proposition's persistence meets a contradictory temporal proposition or its persistence. Suppose that t1 and t2 are tokens with contradictory propositions and one or both of t1 and t2 is persistent toward the other. Then these persistences are clipped, as summarised in the picture below.



In case 1, the forward persistence of P is clipped at t2. In case 2, the backward persistence of ¬P is clipped at t1. In case 3, both persistences are clipped: the forward persistence of P ends and the backward persistence of ¬P begins somewhere between t1 and t2, but we can't say where. Each point between t1 and t2 where clipping may happen represents a different possible world consistent with the domain theory. This is the simplest form of ambiguity resulting from persistence; we discuss others in Section 5.

Using both backward and forward persistence and treating their opposition in this way allows us to handle cases that using forward persistence alone does not. Kauts [Kauts, 1986] describes the "parking lot problem," in which an agent parks a car in a lot, leaves for a time, and then returns finding the car missing, having been stolen. Maximising only forward persistence results in believing that the car was stolen at the last possible instant before the agent returned. Using both backward and forward persistence and accepting each of the possible worlds described for case 3 above results in believing that the car could have been stolen at any time while the agent was gone.

When a proposition persists necessarily to a point using a given theory (*i.e.*, when it is persistent and not possibly clipped with respect to the point), we say that it holds necessarily at that point.

We define $\text{holds}_{\Box}$ and $\text{holds}_{\Diamond}$ as follows. Each definition relies on identifying persistence clippers. $\text{holds}_{\Box}$ relies on determining the absence of weak clippers: if a persistence is not clipped approaching a point in any

[3] For a more careful discussion of the use of model preference for modeling persistence see *e.g.*, [Shoham, 1988].

[4] These formulas always are theorems (*i.e.*, queries), not axioms in the domain theory.

possible world, then it persists until that point in every possible world. $\text{holds}_{\Diamond}$ relies on determining the absence of strong clippers: if a persistence is not clipped in every possible world, then it persists in some possible world. For simplicity, we give definitions for propositional queries, not general query formulas. We define $\text{holds}_{\Box}$ and $\text{holds}_{\Diamond}$ pointwise, relying on the equivalence given above for the point and interval forms of holds(*i.e.*, whatever holds over an interval holds at every point in that interval and *vice versa*).

$$\begin{aligned}&\text{holds}_{\Box}(t, P) \dashv\\ &\quad(\text{holds}_{\Diamond\Box}(t, P)\\ &\quad\vee(\text{persists}_{b}(\textit{begin}, P)\\ &\qquad\wedge\neg\,\text{clipped}_{b\Diamond}(P, \textit{begin}, t))\\ &\quad\vee(\text{persists}_{f}(\textit{end}, P)\\ &\qquad\wedge\neg\,\text{clipped}_{f\Diamond}(P, \textit{end}, t)))\end{aligned}$$

A proposition P holds necessarily at a point t if it holds either by observation necessarily or by persistence necessarily at t. P may hold by observation at t either directly in the domain theory or indirectly from the application of causal rules. $\text{clipped}_{b\Diamond}$ indicates that a backward persistence assumption is clipped possibly before it reaches a point, and $\text{clipped}_{f\Diamond}$ indicates that forward a persistence assumption is clipped possibly.

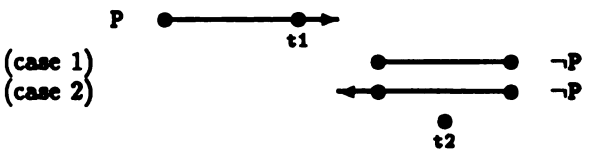
$$\begin{aligned}&\text{holds}_{\Diamond}(t, P) \dashv\\ &\quad(\text{holds}_{\Diamond\Diamond}(t, P)\\ &\quad\vee(\text{persists}_{b}(\textit{begin}, P)\\ &\qquad\wedge\neg\,\text{clipped}_{b\Diamond}(P, \textit{begin}, t))\\ &\quad\vee(\text{persists}_{f}(\textit{end}, P)\\ &\qquad\wedge\neg\,\text{clipped}_{f\Diamond}(P, \textit{end}, t)))\end{aligned}$$

This definition inverts the quantifiers $\Box$ and $\Diamond$ (indicated using subscripts) in the definition for $\text{holds}_{\Box}$.

We define $\text{clipped}_{f\Box}$ and $\text{clipped}_{f\Diamond}$ as follows. The backward cases ($\text{clipped}_{b\Box}$ and $\text{clipped}_{b\Diamond}$) are exactly symmetric.

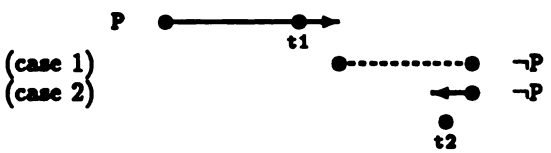
$$\begin{aligned}&\text{clipped}_{f\Box}(P, \texttt{persist-end}, \texttt{point}) \dashv\\ &\quad(\text{holds}_{\Diamond\Box}(t, \neg P)\\ &\qquad\wedge \texttt{persist-end} <_{\Box} t\\ &\qquad\wedge t \leq_{\Box} \texttt{point})\end{aligned}$$

In other words, the observation interval of a clipping proposition must intervene between the endpoint from which the persistence emanates (`persist-end`) and the point of interest (`point`) in every possible world. The picture below shows two cases for which $\text{clipped}_{f\Box}(P, t1, t2)$ is satisfied.



$$\begin{aligned}&\text{clipped}_{f\Diamond}(P, \texttt{persist-end}, \texttt{point}) \dashv\\ &\quad(\text{holds}_{\Diamond\Diamond}(t, \neg P)\\ &\qquad\wedge \texttt{persist-end} <_{\Diamond} t\\ &\qquad\wedge (t \leq_{\Diamond} \texttt{point}\\ &\qquad\quad\vee(\text{persists}_{b}(\textit{clip-end}, \neg P)\\ &\qquad\qquad\wedge\neg\,\text{clipped}_{b\Box}(\neg P, \textit{clip-end}, \texttt{point}))))\end{aligned}$$

In other words, the observation interval of a clipping proposition must intervene between the endpoint from which the persistence emanates and the point of interest in some possible world, or the backward persistence of a clipping proposition must not be clipped itself before it reaches the point of interest. The picture below shows two cases for $\text{clipped}_{f\Diamond}(P, t1, t2)$.



For causal inference, we also need to define $\text{holds}_{\Box}$ and $\text{holds}_{\Diamond}$ for a conjunction of propositions, as found in causal rules.

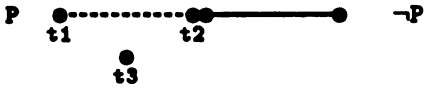
$$\begin{aligned}&\text{holds}_{\Box}(t, (\bigwedge \langle P_1, \ldots, P_k\rangle)) \dashv\\ &\quad(\bigwedge \langle\, \text{holds}_{\Box}(t, P_1), \ldots, \text{holds}_{\Box}(t, P_k)))\end{aligned}$$

$$\begin{aligned}&\text{holds}_{\Diamond}(t, (\bigwedge \langle P_1, \ldots, P_k\rangle)) \dashv\\ &\quad(\bigwedge \langle\, \text{holds}_{\Diamond}(t, P_1), \ldots, \text{holds}_{\Diamond}(t, P_k)))\end{aligned}$$

The version of TMM described in [Dean, 1986, Dean and McDermott, 1987] and originally distributed from Brown University, which we refer to as "α-TMM," does not conform to this semantics. In α-TMM, there was no distinguished observation interval, and persistence was represented by a constraint on a token's endpoints, as in Example 1. Persistence clipping was implemented by constraining persistent propositions further with respect to any contradictory temporal propositions.

Example 1: Persistence in α-TMM.

$\text{persists}_{f}(t2, P)$, so
$t1 \leq t2$



This constraint leaves open the question of whether holds(t3, P) is satisfied—in some total orders it is, and in some total orders it isn't, and the constraint is inadequate to establish necessary persistence. α-TMM used this implementation of persistence to determine when temporal propositions hold, in the predicate tt (for "true throughout"), and it used this tt predicate as the basis for causal projection. Subsequently, Dean and Boddy showed in [Dean and Boddy, 1988] that this implementation is unsound: tt(t, t, P) holds in cases for which there is no possible world in which

holds(t, P).

In our new implementation of TMM, we never represent the "endpoint of a persistence" explicitly; we separate persistence assumptions from observations, and we now use endpoints to represent observation intervals, exclusively. We will refer to our new implementation as "β-TMM." We describe β-TMM more completely in [Schrag *et al.*, 1992].

2.4 INFERENCE

As Dean and Boddy showed in [Dean and Boddy, 1987], reasoning about temporal propositions with causal projection and partial orders by quantifying over the set of consistent total orders, as in $\text{holds}_\Box$ and $\text{holds}_\Diamond$, is an NP-complete problem. We have followed Dean's and Boddy's approach in implementing a decision procedure which approximates the quantification over possible worlds included in our semantics. As in [Dean and Boddy, 1987], the approximation is sound but incomplete (*i.e.*, if the system infers $\text{holds}_\Box(t, P)$, the proposition P does in fact hold at the point t in every possible world, but sometimes this property will be true and the system will not infer $\text{holds}_\Box(t, P)$). Also as in [Dean and Boddy, 1987], the decision procedure executes in polynomial time. We describe this decision procedure in Section 4. The process of implementing it has made clear precisely how the resulting system is incomplete; we address this point in Section 4.

3 SOURCES OF DISJUNCTION

We have explicitly removed one source of disjunction from TMM: we provide no way to assert an explicit disjunction in the domain theory. You can say that proposition P holds at time t, and that point t1 is ordered before point t2. You cannot express the fact that t1 and t2 cannot occur simultaneously (*i.e.*, they definitely are ordered one way or the other), or that either P or Q holds at t3.

This leaves two main sources of disjunction. The first is the temporal uncertainty resulting from unordered time points. The uncertainty from any metric temporal relations that specify the distance between two time points only as a range ultimately is reflected in the ordering of time points, and clipping and projection are determined entirely by that ordering. We give some examples of situations with partial orders resulting in ambiguity in Section 4.

The other main source of disjunction is a direct result of the persistence assumption. Nonmonotonic reasoning has been recognised by many people at many times as a source of ambiguity and unintended conclusions. (Most relevant to our work is Hanks and McDermott's paper on applying nonmonotonic logic to temporal reasoning [Hanks and McDermott, 1986].) Unfortunately, nonmonotonic reasoning appears to be too useful to dispense with. The persistence assumption says that things tend not to change unless something makes them change. If we walk into a room, see that the light is on, and walk out again, it seems both reasonable and useful to conclude that the light was on both before we got there and after we left. Contradictory information (*e.g.*, walking into the room at a later point and noticing that the light is off) will lead us to draw different conclusions. The persistence assumption can lead to ambiguous conclusions in a wide variety of situations, a representative sampling of which we discuss in Section 5.

4 AMBIGUITY RESULTING FROM PARTIAL ORDERS

Inference involving partial orders and projection depends on knowing what facts hold at a given point or over a given interval. The predicate holds (without quantification) is defined only with respect to total orders. For a partially ordered time map we are reduced to determining what facts might hold possibly or necessarily—*i.e.*, in some or all of the total orders consistent with the given partial order—using the predicates $\text{holds}_\Box$ and $\text{holds}_\Diamond$. With even a very simple causal model, this is an NP-complete problem [Dean and Boddy, 1988]. The solution we have implemented in β-TMM is to approximate the necessary quantification, as first presented in [Dean and Boddy, 1987].

β-TMM implements two holds definitions which together provide a sound and incomplete, polynomial-time decision procedure for answering these questions. Each definition approximates a quantification over the possible worlds consistent with the domain theory. holds_s (*strong* holds) is a sound and incomplete approximation to $\text{holds}_\Box$. We use holds_s to identify a subset of all propositions that necessarily hold over a given interval. holds_w (*weak* holds) is a complete and unsound approximation to $\text{holds}_\Diamond$. We use holds_w to identify a superset of all propositions that possibly hold over a given interval.

The following definitions, like those for $\text{holds}_\Box$ and $\text{holds}_\Diamond$, apply to propositional queries, not general query formulas. Note that these are interval-based definitions, not point-based definitions as we used in our semantics; this leads to a problem we discuss below.

$$
\begin{aligned}
&\text{holds}_s(p1, p2, P) \dashv \\
&\quad (\text{holds}_{os}(\textit{begin}, \textit{end}, P) \\
&\quad\quad \wedge (\textit{begin} \leq_\Box p1 \\
&\quad\quad\quad \vee (\text{persists}_b(\textit{begin}, P) \\
&\quad\quad\quad\quad \wedge \neg\, \text{clipped}_{bw}(P, \textit{begin}, p1))) \\
&\quad\quad \wedge (\textit{end} \geq_\Box p2 \\
&\quad\quad\quad \vee (\text{persists}_f(\textit{end}, P) \\
&\quad\quad\quad\quad \wedge \neg\, \text{clipped}_{fw}(P, \textit{end}, p2))))
\end{aligned}
$$

We replace the quantifiers $\Box$ and $\Diamond$ (indicated using subscripts) in the definition for $\text{holds}_{\Box}$ by the "approximate quantifiers" s and w. The remaining definitions (for holds_w, clipped_{fs}, clipped_{fw}, clipped_{bs}, and clipped_{bw}) similarly are in terms of intervals. (Again we omit the clipped_b definitions which are exactly symmetric to those for clipped_f.)

$$\text{holds}_w(p1, p2, P) \dashv$$
$$(\text{holds}_{ow}(\textit{begin}, \textit{end}, P)$$
$$\wedge (\textit{begin} \leq_{\Diamond} p1$$
$$\vee (\text{persists}_b(\textit{begin}, P)$$
$$\wedge \neg \text{clipped}_{bs}(P, \textit{begin}, p1)))$$
$$\wedge (\textit{end} \geq_{\Diamond} p2$$
$$\vee (\text{persists}_f(\textit{end}, P)$$
$$\wedge \neg \text{clipped}_{fs}(P, \textit{end}, p2))))$$

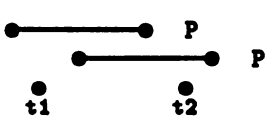
$$\text{clipped}_{fs}(P, \texttt{persist-end}, \texttt{point}) \dashv$$
$$(\text{holds}_{os}(\textit{begin}, \textit{end}, \neg P)$$
$$\wedge \texttt{persist-end} <_{\Box} \textit{begin}$$
$$\wedge \textit{begin} \leq_{\Box} \texttt{point})$$

$$\text{clipped}_{fw}(P, \texttt{persist-end}, \texttt{point}) \dashv$$
$$(\text{holds}_{ow}(\textit{begin}, \textit{end}, \neg P)$$
$$\wedge \texttt{persist-end} <_{\Diamond} \textit{begin}$$
$$\wedge (\textit{begin} \leq_{\Diamond} \texttt{point}$$
$$\vee (\text{persists}_b(\textit{clip-end}, \neg P)$$
$$\wedge \neg \text{clipped}_{bs}(\neg P, \textit{clip-end}, \texttt{point}))))$$

In our implementation we retain sound and complete (not approximate) temporal relation inference. All of the temporal propositions (observations) in the domain theory hold strongly; the results of causal rules whose antecedents (and, for projection rules, the trigger event) hold strongly also hold strongly; the results of those whose antecedents hold weakly also hold weakly.[5]

The fact that these definitions are interval-based rather than point-based as in our semantics means that the approximation always requires a single token to span the whole query interval. Because of this, holds_w will not succeed in some cases where it should, according to the semantics we define in Section 2.3. This makes an inference method which we have said must be "complete" incomplete. Consider the following example.

Example 2: holds_s is incomplete because it considers only single tokens.



P covers the interval from t1 to t2 in all possible worlds, so $\text{holds}_{\Box}$(t1, t2, P). However, with the single-token (interval) definition for holds_s, we do not believe this (even weakly).

Our planned solution is to modify the definition of holds_w (and holds_s) to reason fully about overlapping tokens. The will make both approximations—holds_w and holds_s—as accurate as possible, benefiting the inferential power of the system overall.

We now are prepared to characterise the incompleteness of holds_s and the unsoundness of holds_w.

The definition of holds_s relies on strong tokens (observations), necessary temporal relations, and in the case of persistence the absence of weak defeaters. holds_s is incomplete in two ways.

- It avoids combinatorics by looking for a single token (or a single set of tokens) to span the query interval for all possible worlds. It will fail in a case where the interval is spanned by different tokens (or sets) in different possible worlds.
- It relies, ultimately, on the over-achieving holds_w to defeat the strong tokens' persistences.

The definition of holds_w relies on weak tokens, possible temporal relations, and in the case of persistence the absence of strong defeaters. holds_w is unsound in two ways.

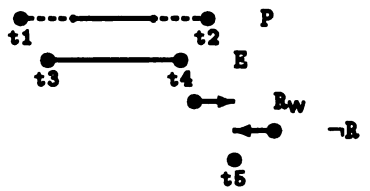
- It avoids combinatorics by checking for a conjunction of possibilities rather than a possible conjunction. It succeeds sometimes when the conjuncts are not mutually satisfiable.
- It relies, ultimately, on the under-achieving holds_s to defeat the weak tokens' persistences.

We note that making the strong inference component of our system more incomplete probably doesn't make inference as a whole less expensive. The weak component must be complete for the strong component to be sound, and the (weak) results from weak causal inference proliferate, creating more work. Also, less complete results from strong inference mean less defeaters for weak inference, and more weak results. There probably is a critical point at which weak defeaters overwhelm strong inference and make persistence practically useless. The degree of proliferation of equally plausible possible worlds is problem-dependent, and we warn users about the sources of complexity in our system, as well as doing as much as we can to limit them.

[5] All of the tokens in our time map pictures hold strongly unless otherwise indicated.

Example 3: Unsoundness in $holds_W$ can arise directly from partially ordered timepoints, or indirectly through projection.

distance(t1, t2, 3)
distance(t3, t4, 5)
project(P, E, R)

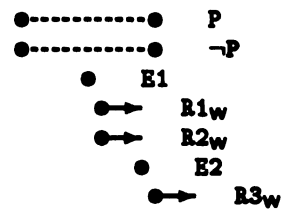


P does not cover E in any possible world, so we do not have $holds_\Diamond$(t3, t4, P)—but the conjunction of possible temporal relations in $holds_W$(t3, t4, P) is satisfied, and it succeeds creating R_W, unsoundly. The source of this problem is that while $t1 \leq t3$ and $t2 \geq t4$ both are possible, their conjunction is not.

This example also shows how the unsoundness of $holds_W$ contributes to the incompleteness of $holds_S$ through weakly and unsoundly derived defeaters such as R_W. We should have $holds_\Box$(t5, ¬R), but ¬R is clipped weakly and unsoundly, and we are limited to $holds_W$(t5, ¬R).

Example 4: Contradictory tokens without any ordering information result in unsound derivations.

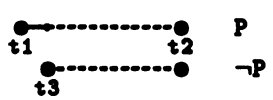
project(P, E1, R1)
project(¬P, E1, R2)
project((R1 ∧ R2), E2, R3)



Although P and ¬P never are believed in the same possible world, both are believed weakly covering E1. Both $R1_W$ and $R2_W$ are derived soundly, but $R3_W$ is derived unsoundly, because R1 and R2 actually never occur in the same model.

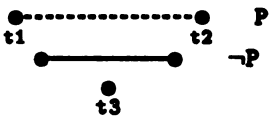
Example 5: Pairwise contradictory tokens with some ordering information can be caught.

$distance_\Box$(t1, t3, 20)



When t1 and t3 are ordered, we can then add the constraint $t2 < t3$, and P no longer is believed weakly after t2 or at t3, thus improving the approximation of $holds_W$ to $holds_\Diamond$.

Example 6: Unordered contradictory tokens where one is necessarily true can be caught.



We can strengthen the definition of $holds_W$ by modifying it so that $holds_W$(t3, P) fails if $holds_S$(t3, ¬P) succeeds.

5 AMBIGUITY RESULTING FROM PERSISTENCE

The persistence assumption can be thought of as a preference over the set of models (possible worlds) satisfying a given domain theory: we prefer models in which propositions persists as long as possible, both backward and forward in time. When different, equally plausible models exist for a given theory (none of which is preferred over all the others) ambiguity results. These ambiguities arise even in theories where all temporal relations are precisely specified for every point in the time map, as they are for all of the examples in this section.

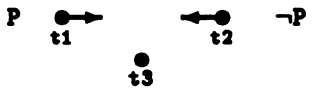
The ambiguities resulting from persistence that we discuss in this section fall into three main categories.

- Ambiguity from mutually clipping propositions.
- Ambiguity from cycles of causal rules.
- Ambiguity from clipping overlap chaining antecedents on the basis of contradicted results.

5.1 AMBIGUITY FROM MUTUALLY CLIPPING PROPOSITIONS

The simplest kind of ambiguity from persistence results when two contradictory temporal propositions persist in opposite directions, as discussed in Section 2.3. Besides the direct ambiguity over where the persistences clip, as in Example 7 below, such ambiguity also can result indirectly through the effects of causal rules, as in Example 8.

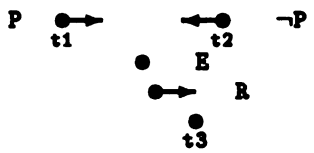
Example 7: Ambiguity from mutually clipping propositions results in approximation.



In this example, P and ¬P clip one another, somewhere between t1 and t2, but we cannot determine where. In particular, we do not know which of them holds at t3. We approximate the disjunction by saying that both of them hold weakly.

Example 8: Incompleteness in $holds_S$ can arise from opposing contradictory persistences.

project(P, E, R)
project(¬P, E, R)



Our semantics says that the persistences for P and ¬P clip at some point between t1 and t2, but not where. One of P or ¬P covers E in all possible worlds, so $holds_{\Box}(t3, R)$ is satisfied using the projection rules. The strongest inference that TMM can make is $holds_W(t3, R)$.

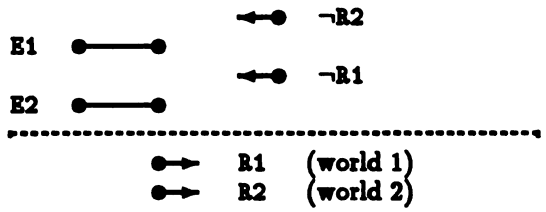
5.2 AMBIGUITY FROM CYCLES OF CAUSAL RULES

More complicated ambiguities arise when we consider the interaction of persistence with causal rules involving cycles, in which an eventual consequent undermines one of the original antecedents. Only one of the antecedent or the offending, contradictory consequent holds in any possible world, but neither is preferred by our semantics.

In the examples below we address this problem by introducing "weak clipping" for persistences undermining causal rule antecedents. In projection, we weakly clip the backward persistence of a token matching a rule result that possibly would undermine the antecedent of another rule, at the point where it meets the observation interval of a token that matches the undermined rule's event. In overlap chaining, we clip persistences as soon as they meet the observation interval of a token that matches any of the the undermined rule's antecedents. In either case, until meeting one of these weak clippers, the persistence holds strongly, if it is not clipped by another token. We describe the action of this weak clipping approach, in which we again use a weaker form of representation to approximate a disjunction, for both kinds of causal reasoning in the examples below. It is conservative and could be considered rather heavy-handed, as we will discuss.

Example 9: Projection with backward persistence can be ambiguous.

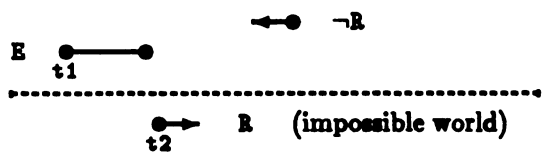
project(¬R2, E1, R1)
project(¬R1, E2, R2)



Each projection rule defeats the other's antecedent. In possible world 1, ¬R2 and R1 are believed; in possible world 2, ¬R1 and R2 are believed. Our incremental algorithm would fire strongly whichever rule it happened to see first, making it unsound. Weak clipping will ensure that both results are derived weakly. The reason this is heavy-handed is that we have to clip the contradictory persistences as soon as we see the trigger event, whether the other antecedents are on the time map or not, against the eventuality that they will be asserted.

Example 10: A single projection rule, while not ambiguous, may have a difficult semantics to implement.

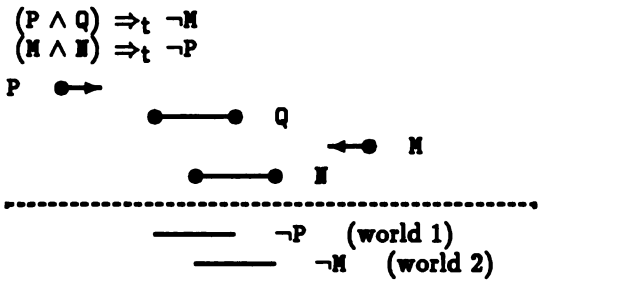
project(¬R, E, R)



The projection rule should never fire because the result defeats the antecedent. Our semantics prefers a world in which ¬R persists back just until t1 but not to it. For the rule to fire there must be some, as yet unknown, defeater for ¬R that occurs after t2; we acknowledge an incompleteness in the domain theory. Weak clipping will fire the rule weakly, which is unsound, but that may be the best we can do.

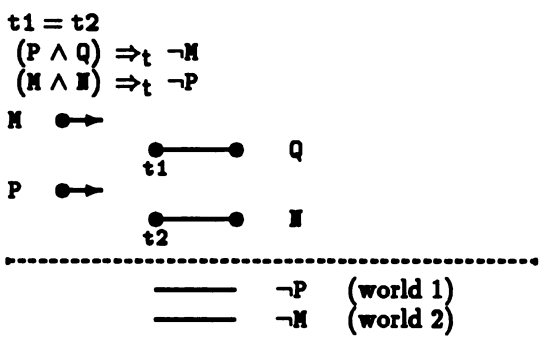
In the examples involving overlap chaining below, we represent the result propositions pictorially as lines without endpoints: ————. (This is because they are present on the time map only implicitly.)

Example 11: Overlap chaining with persistence can be ambiguous.



Each overlap chaining rule defeats one of the other's antecedents. In possible world 1, ¬P is believed and P is clipped; in possible world 2, ¬M is believed and M is clipped. Weak clipping acts much as in Example 10, weakening persistence just before it one of the antecedents. This is heavy-handed because we clip contradictory persistences as soon as the antecedents appears.

Example 12: Overlap chaining with just forward persistence can be ambiguous.



This is like example 11 but shows that the problem is not an isolated result of our introducing backward persistence into β-TMM.

Besides weak clipping, which introduces approximation in a rather heavy-handed way, we can handle this problem by detecting when a causal theory has cycles and try to prevent these kinds of ambiguities. We do this by inspecting a directed acyclic graph (dag) connecting rule antecedents to rule consequents. The connections are made whenever antecedents unify with consequents (rule chaining) or consequents' negations (rule antecedent undermining). Whenever there exists a cycle in the dag with at least one undermining connection, there is a potential for ambiguity. For this potential to be realised, one of the antecedents of the original rule must be satisfied using persistence, and a contradictory consequent must intervene temporally to defeat the original antecedent.

With this static rule analysis technique, we have identified are two possible approaches to preventing this kind of ambiguity.

- Reject rule sets that may result in it.
- Instead of rejecting rules, prohibit backward persistence for tokens that unify with undermining results. (This algorithm currently is implemented in β-TMM.)

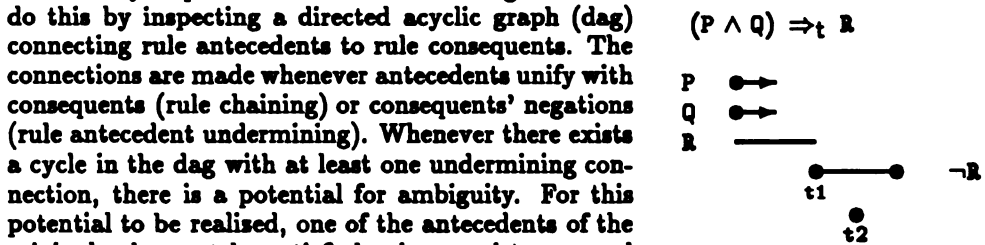
Either of these prevention approaches is sound, but both limit functionality. Including the weak clipping approach, all three overreact to prevent situations that may never occur on the grounds that detecting these specific situations is too expensive—we believe this would require a non-incremental algorithm for firing rules, which is impractical. All three approaches handle the ambiguous examples above.

5.3 AMBIGUITY FROM CLIPPING OVERLAP CHAINING ANTECEDENTS ON THE BASIS OF CONTRADICTED RESULTS

We view overlap chaining as syntactic substitution. We do not make overlap chaining results explicit as time tokens on the time map, and they are present only implicitly. We need to view overlap chaining as syntactic substitution for the purpose of clipping also. *E.g.*, consider a rule $(P \wedge Q) \Rightarrow_t R$. Since we have clips(¬R, R), we also must have clips(¬R, (P ∧ Q)). That is, we need to clip belief in the conjunction (P ∧ Q).

When this conjunction holds strongly by strong persistence of both antecedents, we can use a weak clipping approach similar to that described above for cycles of causal rules. We apply weak clipping where the antecedents' persistence meets a contradictory observation. Consider the following example.

Example 13: Clipping multiple strong overlap chaining antecedents on the basis of a contradictory observation results in ambiguity.



We clip both antecedents weakly beginning at t1, avoiding a situation in which $holds_s(t2, (P \wedge Q))$ but not $holds_s(t2, R)$. Nonetheless, we still do have $holds_w(t2, R)$; that is, clipping the antecedents effectively clips the result also.

6 EXPLICITLY REPRESENTING DISJUNCTION USING CONTEXTS

Asserting temporal propositions weakly in the data base as we do in our approximate decision procedure is tenable when the temporal propositions are intended to serve as a very conservative filter to limit beliefs that are defeasible; it really is not valuable in itself for deriving positive conclusions. Within the TMM system we have applied this weak representation exclusively to the results of causal reasoning, and we do not suggest that users attempt to manage explicit disjunctions of temporal propositions in their problems by asserting each of the disjuncts as a separate weak token. With our weak representation, all distinction among alternatives that may be incompatible is lost, as all individually valid contexts are combined into a single, generally invalid context. Keeping compatible alternatives separate requires a mechanism for managing genuinely separate contexts.

We are contemplating a context management facility for TMM, in which a context would be a transaction stream recording a user's changes to the data base. Users could branch from any point in a transaction stream to create subcontexts, and we would maintain these branching points to enable context-switching. Switching between two such contexts might be expensive, though, because TMM might need to do a significant amount of inference to account for the transactions executed since the contexts' most recent common branching point. The responsibility for creating and switching among these contexts rests in the hands of users, who must be careful to avoid combinatorial explosion of disjunctive alternatives in their problems and unproductive context thrashing.

7 RELATED WORK

The generation and management of ambiguity in causal and temporal reasoning has been investigated in several domains, using various approaches, most explicitly in recent work by Stein [Stein, 1990].

Recent theoretical work has characterised the complexity of reasoning with temporal algebras of various degrees of expressivity. Vilain *et al.* [Vilain *et al.*, 1989] deal with Allen's interval algebra [Allen, 1983] and two restrictions of it to point algebras, all of which are non-metric. Dechter *et al.* [Dechter *et al.*, 1991] deal with a metric point algebra, with constraints over both convex and non-convex intervals. Our graph of points and constraints over convex intervals is conceptually an instance of their "simple temporal problem" (STP), for which inference about temporal relations has worst-case cubic complexity. As Dechter *et al.* point out, the general version of this problem with constraints over non-convex intervals (the "temporal constraint satisfaction problem," or TCSP) is NP-hard. This work does not address the issues that arise from mapping propositions onto intervals of time in a temporal data base system, or persistence and causal reasoning.

Since the introduction of TMM [Dean, 1986], other researchers have recognised the importance of reasoning about time "from the side" in a temporal data base management system, and have developed various other "time map managers" (generically speaking) involving points or intervals, constraints, and partial orders [Materne and Hertsberg, 1991, Ghallab, 1989, Koomen, 1988, Muscettola, 1990, Drabble and Kirby, 1990]. The authors of MTMM [Materne and Hertsberg, 1991] seek to correct the confusion of persistence and temporal uncertainty in α-TMM (mentioned in Example 1), but their solution is not as general as ours, and their implementation was intended only as a prototype. TMM is set apart from all of these systems by its stronger notion of persistence and its sound and incomplete algorithm for temporal projection. This orientation has given us our unique approach to managing disjunction.

Other authors [Nebel and Backstrom, 1992, Bylander, 1991] have referred to the work of Dean and Boddy [Dean and Boddy, 1988] in discussing the complexity of various restricted classes of planning problems. However, none of these researchers have directly addressed the same problem or contributed to a solution to the specific problem of causal reasoning with partially ordered events. To our knowledge, β-TMM is the first working implementation of sound polynomial temporal reasoning with partially ordered events.

8 SUMMARY

In this paper, we have identified sources of disjunction that must be considered in a temporal reasoning system that handles partially ordered time points, forward and backward persistence, and two simple forms of causal reasoning. These sources can be grouped roughly into two classes: one corresponding to problems arising from temporal uncertainty (partial orders); and the other arising from the nonmonotonic persistence assumption. So far we have avoided a third source of disjunction by restricting the expressive power of the system—we do not permit the expression of explicit disjunctive formulas in the domain theory.

We have demonstrated the application of a particular approach to dealing with disjunction that uses a weak representation that subsumes the set of disjuncts. This weak representation has the advantage that it can be computed efficiently, and the obvious disadvantage that the system is now incomplete, though still sound. This paper presents the first clear characterisation of the sources of incompleteness in this approach, first

introduced in [Dean and Boddy, 1988].

The techniques we have developed for managing disjunction are crucial to our implementation of an efficient temporal reasoning system. In particular, the representation of a set of disjunctions by some simpler description of a larger set including those disjunctions is a powerful technique that has found repeated use for handling disjunctions with a wide variety of sources and characteristics. With a little care, the resulting system will retain the property of soundness, which we regard as crucial to the implementation of a practical system for temporal reasoning.

Acknowledgements

This work is supported by DARPA and the Air Force Rome Laboratory under Rome Laboratory contract F30602-90-C-0102.

References

[Allen, 1983] Allen, James 1983. Maintaining knowledge about temporal intervals. *CACM* 26(11):832–843.

[Bylander, 1991] Bylander, Thomas 1991. Complexity results for planning. In *Proceedings of the Twlefth International Joint Conference on Artificial Intelligence (IJCAI-91)*. 274–279.

[Dean and Boddy, 1987] Dean, Thomas and Boddy, Mark 1987. Incremental causal reasoning. In *Proceedings AAAI-87 Sixth National Conference on Artificial Intelligence*. 196–201.

[Dean and Boddy, 1988] Dean, Thomas and Boddy, Mark 1988. Reasoning about partially ordered events. *Artificial Intelligence* 36(3):375–399.

[Dean and McDermott, 1987] Dean, Thomas and McDermott, Drew 1987. Temporal data base management. *Artificial Intelligence* 32(1):1–55.

[Dean, 1986] Dean, Thomas 1986. *Temporal Imagery: An Approach to Reasoning about Time for Planning and Problem Solving*. Ph.D. Dissertation, Yale University.

[Dechter *et al.*, 1991] Dechter, Rina; Meiri, Itay; and Pearl, Judea 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.

[Drabble and Kirby, 1990] Drabble, Brian and Kirby, R. 1990. Associating AI planner entities with an underlying time point network. Technical report, Artificial Intelligence Applications Institute, University of Edinburgh, Scotland.

[Ghallab, 1989] Ghallab, Malik 1989. Managing efficiently temporal relations through indexed spanning trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*. 1297–1303.

[Hanks and McDermott, 1986] Hanks, Steve and McDermott, Drew 1986. Default reasoning, nonmonotonic logics, and the frame problem. In *Proceedings AAAI-86 Fifth National Conference on Artificial Intelligence*. 328–333.

[Kautz, 1986] Kautz, Henry 1986. The logic of persistence. In *Proceedings AAAI-86 Fifth National Conference on Artificial Intelligence*. 401–405.

[Koomen, 1988] Koomen, Johannes 1988. The Timelogic temporal reasoning system. Technical Report 231 (revised), University of Rochester Computer Science Department.

[Materne and Hertsberg, 1991] Materne, Stefan and Hertsberg, Joachim 1991. MTMM: Correcting and extending time map management. Technical Report Arbeitspapiere der GMD 511, GMD.

[Muscettola, 1990] Muscettola, Nicola 1990. Integrating planning and scheduling to solve space mission scheduling problems. In *Proceedings DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control*. Morgan Kaufmann. 220–230.

[Nebel and Backstrom, 1992] Nebel, Bernhard and Backstrom, Christer 1992. On the complexity of temporal projection and plan validation. In *Proceedings AAAI-92 Tenth National Conference on Artificial Intelligence*. 748–753.

[Schrag *et al.*, 1992] Schrag, Robert; Carciofini, Jim; and Boddy, Mark 1992. β-TMM manual. Technical Report CS-R92-012, Honeywell SRC.

[Shoham, 1987] Shoham, Yoav 1987. Temporal logics in AI: Semantical and ontological considerations. *Artificial Intelligence* 33(1):89–104.

[Shoham, 1988] Shoham, Yoav 1988. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press.

[Stein, 1990] Stein, Lynn 1990. *Resolving Ambiguity in Nonmonotonic Reasoning*. Ph.D. Dissertation, Department of Computer Science, Brown University. Available as TR CS-90-18.

[Vilain *et al.*, 1989] Vilain, Marc; Kautz, Henry; and van Beek, Peter 1989. Constraint propagation algorithms for temporal reasoning: A revised report. In de Kleer, Johan and Weld, Daniel, editors 1989, *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufmann. 373–381.

Infinite Loops in Finite Time: Some Observations

Ernest Davis
Courant Institute
New York, New York 10012

Abstract

Formulating physical theories using real-valued space and time often raises the difficult issue of *clustered variation:* states that change their value infinitely often in a finite interval, or infinite sequences of events that occur in a finite interval. Some physical theories permit nonsensical models unless clustered variation is prohibited; others force clustered variation to occur. This paper surveys a number of technical issues involved with clustered variation. We present examples of theories where clustered variation must be allowed and of theories where it must be prohibited. We discuss the meaning of plans containing infinite series of events. We present constraints on real-valued fluents and spatial fluents that guarantee discrete variation of important associated states. We discuss the dependence of this issue on the choice of ontology.

1 INTRODUCTION

Viewed from a modern perspective, the great insight in Zeno's paradoxes is that common sense is confused about the infinite divisibility of space and time. On the one hand, it seems absurd to suppose that two instants or two points can be distinct and yet have no instant or point between them. On the other hand, if space and time are dense, then the arrow trying to reach its goal or Achilles trying to catch up with the tortoise must execute infinitely many movements, which does not seem right either.

The mathematical problems here have been pretty well settled since Zeno's time. For most purposes, it is agreed, space and time can be modelled as $\Re^n$. Thus, it is consistent to say both, "Achilles executes the action, 'Go to the current position of the tortoise' infinitely many times," and "Achilles catches up to the tortoise in finite time." This theory leads to paradoxes even stranger than Zeno's: the expression of a discontinuous function as the infinite sum of continuous functions; nowhere-differentiable continuous curves; the Banach-Tarski paradox; and so on. Some of these, such as Fourier series, are accommodated within the language of physics; others, such as the Banach-Tarski paradox, are considered mathematical constructs without physical realization. The dividing line between the two is generally left to the physical intuitions of the human understander. These intuitions change over time, of course. A century ago, fractal curves were mathematical monstrosities; today, they are an important element of our descriptive language.

These problems arise anew in trying to characterize commonsense domains for automated reasoners. A domain theory that uses real-valued time must prescribe which of the mathematically possible behaviors can, in fact, occur. Particularly knotty problems are associated with *clustered variation* or *chatter*: a Boolean fluent that change its truth-value infinitely often in a finite time interval, or an infinite sequences of events that occurs within a finite time interval. For example, should a theory allow a light to be on during the interval $t \in [0, 1/2)$, off during $t \in [1/2, 3/4)$, on during $t \in [3/4, 7/8)$, off during $t \in [7/8, 15/16)$ ...? Should it allow the execution of an infinite loop in which the first iteration is executed in 1/2 second, the second in 1/4 second, the third in 1/8, and so on?

Some researchers (e.g. (Hayes 1985), (Fleck 1988), (Kaufmann 1991)), noting that such clustering can neither be perceived, due to the limited resolution of perception, nor actually occur, due to the quantization of small-scale physics, have suggested that $\Re^n$ is a poor model for space and time, and that another topology should be found. However, the suggested alternatives lead to constructs as peculiar as those encountered in $\Re^n$ (e.g. the definition of a "straight line" in (Kaufmann 1991)) and there are basic physical phenomena, such as rotation, that have not received any adequate treatment in these theories. (Fleck's proposed topology may possibly be an exception, but it is not clear how it can be used in physical theories.) Therefore,

the suggestion of alternative topologies must be taken at present as conjectural, I believe. The motivations cited above are not promising leads toward better theories of space and time. Limited perceptual resolution is better treated as an aspect of perception than as an inherent feature of space and time (Davis 1989a). The quantum theory is very far from a commonsense understanding.

Alternatively, we could use real-valued space and time, but impose a regularity condition. One such condition was formulated (by me) as axiom 9 of (McDermott 1982). A Boolean fluent *varies discretely* if it never has clustered variation; that is, if it changes its value only finitely many times in any bounded time interval. The "axiom of discrete variation" posits that every Boolean fluent varies discretely.[1]

"The method of 'postulating' what we want has many advantages; they are the same as the advantages of theft over honest toil." (Russell 1919). Imposing the axiom of discrete variation creates two serious difficulties. First, the axiom of discrete variation can only hold if restrictions are placed on the the language of states, the behavior of real-valued fluents, and the shapes, motions, and placements of objects. For example, "the temperature of the tea being a rational number" must be excluded as a possible state. The function $x(t) = t \sin(1/t)$ must be excluded as a possible behavior, if the condition $x(t) > 0$ is to be allowed as a fluent. This issue was discussed in (McDermott 1982), but not in depth.

The second difficulty is that a number of physical theories actually predict the occurrence of clustered variation. These theories are idealizations, but they are reasonable and useful idealizations, which can only be avoided at substantial cost. The best known of these (discussed in (Fleck 1988)) is the bouncing ball (Figure 1). The simplest model of a ball bouncing on the ground is that the ball is a rigid object, and that at each bounce, the ball bounces up with a speed that is a fixed fraction μ of the speed with which it hit. Since the time between bounces is proportional to the speed leaving the ground, it follows that the time between bounces decreases in a geometric series, and that therefore the ball attains perfect rest in a finite time, having carried out infinitely many bounces. (Note that this is different from a standard damped harmonic oscillator, in which the frequency is constant, and the system therefore in principle never reaches a state of absolute rest.) Admittedly, this is an idealization; with an actual ball, beyond a certain point, the system will be dominated by the internal vibrations of the ball, which have a fixed frequency. However, to take this into account would require modelling the ball as an elastic object, which greatly complicates reasoning.

[1] A similar but weaker rule was proposed in (Hamblin 1971). (Galton 1987) discusses some philosophical treatments of clustered variation.

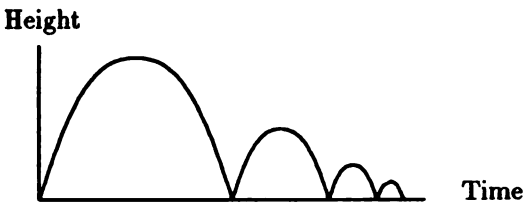


Figure 1: Bouncing Ball

This scenario, though peculiar, does not constitute an *inconsistency* in the Newtonian theory of rigid solid objects. The position and the velocity of the bouncing ball approach a unique limit as the bounces go to zero, so its state after the infinite series of bounces is well-defined and easily categorized axiomatically.

One might propose that a reasoning system should treat the ball as rigid except at times where it is predicts clustered variation, where it should use a more realistic model. The problem is that in a general reasoning system, the prediction of clustered variation may be subtly hidden. Consider, for example, the following problem: Given that a ball is at rest on a surface at time t, is it possible that the ball was bouncing any time before t? One can easily imagine (or build) a program that constructs the following chain of reasoning:

> Suppose that the ball was bouncing before t. Consider the last bounce B. If the ball attained height x on B, then it would have hit table with velocity $v = \sqrt{2gx}$. But then it would have bounced up again with velocity μv, which contradicts the assumption that B was the last bounce. Therefore, the ball has always been at rest on the table.

Note that the conclusion is not obviously absurd; there is nothing that would force the withdrawal of a default assumption, for example. Note also that the assumption of discrete variation is "hidden" in the assumption that if the ball bounced, then there was a last bounce. Neither the program nor its creator need ever do any thinking about infinite sequences of events. Put the point another way: a qualitative simulator, addressing this system, would generate an envisionment graph in which there was no transition from a bouncing state to a rest state (Figure 2). Though the graph is correct, it does not justify the conclusion that a ball that is now bouncing will never be at rest, or that a ball that is now at rest was never bouncing.

Thus, it was overkill to apply the axiom of discrete variation to all fluents. A more reasonable approach is to define "discrete variation" as an interesting property, and then to posit that specific key fluents have this property. The problem, then, is to determine which fluents can and should be so constrained.

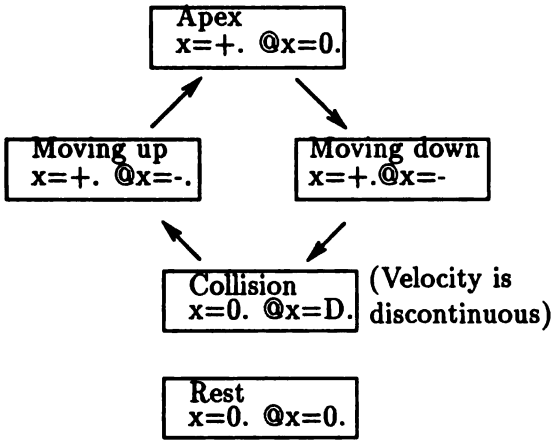


Figure 2: Envisionment of Bouncing Ball

For several years, I have worked on formal characterizations of physical theories using real-valued space and time. In each domain, the issue of clustered variation has arisen in new ways and created new, peculiar problems. In my experience, there are no general-purpose regularity conditions that can be imposed on all theories of this kind. Rather, the issue must be thought through separately for each theory, and for each fluent in the theory. The regularity conditions to be imposed depend on the range of physical phenomena considered; the idealizations adopted; the kinds of fluents considered as first-class entities; and the treatment of other issues of space and time.

Accordingly, this paper does not attempt a general discussion of clustered variation. Rather, it addresses a number of technical issues that arise. Section 2 discusses why clustered variation should sometimes be prohibited. Section 3 presents further examples of natural theories that generate clustered variation. Section 4 discusses infinite sequences of events. Section 5 presents conditions on the types of states, real-valued fluents, shapes, and motions that are sufficient to guarantee discrete variation. Section 6 discusses how the necessity or impact of requiring discrete variation may differ depending on how the world is carved up into entities. Because of space limitations, some proofs are omitted; they are found in (Davis 1991b).

2 WHY EXCLUDE CLUSTERED VARIATION?

In (McDermott 1982) the axiom of discrete variation is motivated by the following example. Suppose we wish to prove the following theorem: "If day is always followed by night, and night is always followed by day, and it is now day, then it will always be either day or night." This theorem cannot be proved because it

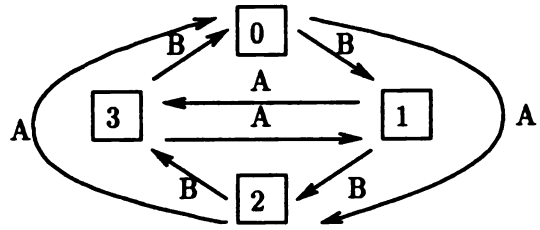


Figure 3: Transition Network

is not logically necessary. The following scenario is consistent: Day from t=0 to t=1/2; then night from t=1/2 to t=3/4; then day until t=7/8; then night until t=15/16 ... and, starting at t=1, a state that was neither day nor night. In such a case, day would always be followed by night and night by day, but eventually it would be neither day nor night. (Compare, "If each bounce of the ball is followed by another bounce, then the ball will always be bouncing," which may be false.) However, the proof goes through if we posit that "day" and "night" vary discretely.

A sceptic, however, could claim that this is a weak argument for requiring discrete variation. Suppose that the conclusion, "It is always day or night" does not follow from the premise "Day is followed by night and night by day," what harm is there in that gap? Why should one want to perform such an inference at all? After all, the statement "It is always day or night," is as reasonable an axiom, with as much direct evidence, as "Day is followed by night, and night by day."

We therefore modify the example. Consider a finite state system with four states numbered 0, 1, 2, 3; an action A, which adds 2 to the state, mod 4; and an action B, which adds 1 to the state, mod 4 (Figure 3). Now, given that the system is in state 0 in situation S1 and that only action A occurs between S1 and S2, infer that at S2 the system in an even-numbered state. This may be a useful thing to infer. For example, if you are playing a game against an opponent and you wish to keep the game state out of odd-numbered states, it is useful to know that you can do this indefinitely, as long as you can prevent the occurrence of any actions other than A. Another example: if the states are nodes in an envisionment graph from a qualitative physical reasoner, such as QP (Forbus 1985), then the usual interpretation of the graph includes the inference that the system will always be in a state that can be reached on a path through the graph.

The problem is that the result does not follow, if clustered variation is possible. It is consistent with the theory that, if action A occurs once over the first half of the interval, and then a second time over the next quarter, and a third time over the next eighth ... then

at time S2, after infinitely many iterations of A, the system should be in either state 1 or state 3. And this is not due to a gap in the logic; this really can happen in systems where clustered variation is possible, such as the bouncing ball. Any single transition "collision with the ground" preserves the state "ball is moving" but the infinite sequence of these transitions does not.[2]

Clearly, we cannot impose an axiom of the form "The system must be in an even-numbered state," analogous to the axiom "It is always day or night;" the system is allowed to be in odd-numbered states at times before S1 or after S2. One approach would be to add a frame axiom, "The system cannot change from the parity of the state between S1 and S2 unless an action of type B occurs." But such an approach could easily lead to a horrific version of the frame problem. One can easily construct networks in which each different subset of actions generates a different partitioning of the network into strongly connected components. In such a network, one would need a different frame axioms for each such set of actions; $2^k - 1$ axioms for k different actions. Since all these axioms are computable from the network description, this is preposterously inefficient. The only reasonable approach, therefore, is to demand that the states of the network vary discretely, in which case the conclusion follows directly.

A second example: Suppose we posit the following axioms:

- Given that window W is in good repair in situation $S1$, it will be broken in situation $S2$ if and only if some object has hit it in the interval $(S1, S2]$.
- Object O is responsible for breaking window W if and only if there is a situation S such that W is in good repair up to S and O hits W in S.

We would like to infer the rule that, if W goes from being in good repair to being broken, then some object is responsible for breaking it. Again, this is not a valid inference. Consider the scenario where the window is hit by some object at time $t = 1$; before that, it had been hit at time $t = 1/2$; before that, at time $t = 1/4$ ...Then certainly the window is broken at any time $t > 0$. However, since each hit occurs after it is already broken, no object is responsible for breaking it.

3 OTHER EXAMPLES OF CLUSTERED VARIATION

Within reasonable theories of mundane physical domains, I have found three types of examples of clustered variation: bouncing rigid solid objects, idealized reflected light, and oscillators with a varying control parameter.

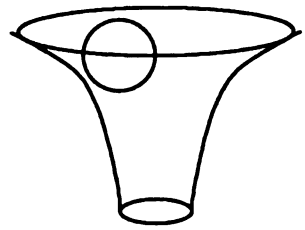


Figure 4: Ball in Funnel

Bouncing: Our original example above of clustered variation was a ball bouncing to heights that reduce in geometric progression. Other more complex examples can be contrived: (1) The restoring force need not be gravity; a ball bouncing off a surface, and returned by an electric field or a spring will (in principle) behave the same way. (2) A ball falling through a smoothly curved funnel whose mouth is just as large as the ball and has a vertical tangent will have infinitely many bounces (Figure 4). (3) Consider a scenario where a large block of mass M is moving towards a fixed wall, and a small ball of mass m is bouncing between them with a coefficient of restitution μ. Suppose that, initially, the velocity of the block is v and the velocity of the ball is u (Figure 5). Let μ, m, and u be small as compared to M and v; specifically, let the following two inequalities be satisfied

$$(1-\mu)^2 \geq 4\mu m/M$$

$$\frac{v}{u} \geq \frac{(1-\mu) - \sqrt{(1-\mu)^2 - 4\mu(m/M)}}{2\mu}$$

Then the ball will bounce between the wall and the block infinitely often as the block converges in a finite time to a distance from the wall exactly equal to the diameter of the ball. The proofs of (2) and (3) are given in (Davis 1991b).

Reflected light: Consider a rectangular box of dimensions $p \times q$ whose internal sides are mirrors. A ray of light is released from one corner of the box with angle θ. The coordinate c_k where the ray hits the left wall for the kth time can thus be calculated as follows: Let $l_k = 2kp \tan\theta$. If $\lfloor l_k/q \rfloor$ is odd, then $c_k = -l_k \bmod 2q$; else $c_k = l_k \bmod 2q$. However, if $p \tan\theta/q$ is irrational, then no two values c_i and c_j are equal; hence the points c_k are densely scattered along the line. (Figure 6)

If we modify the model to use a beam of limited width and uniform intensity and imperfect mirrors that reduce intensity by a fixed fraction, then the whole internal surface will be lit up after finitely many reflections;

[2]This inference may also fail in envisionment graphs derived from theories with quantities of infinitely different orders of magnitude. (Davis 1989b, p. 429)

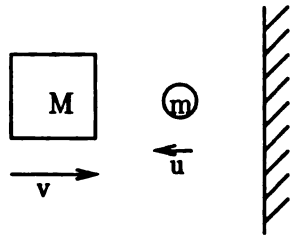


Figure 5: Ball bouncing between Moving Mass and Wall

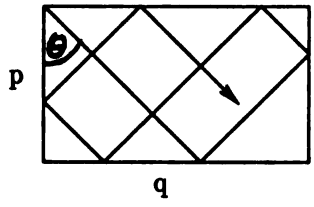


Figure 6: Reflected ray of light

however, the intensity of illumination will be a fractal function of position.

This clustered variation is spatial rather than temporal, but, as we shall see further in section 5.3, the two are closely related. If an object O is moving along the side of the box, then the state "The front point of O is illuminated" in the first case, or the state "The total illumination of O is greater then K" in the second case, may have clustered variation. The idealization breaks down, of course, when we consider the finite wavelength of light, the quantum nature of light intensity, or the finite speed of light, but in ordinary applications, it is convenient to idealize away all of these.

Variable frequency oscillation: Most physical oscillators behave, at least around their equilibrium point, like damped harmonic oscillators; that is, they oscillate with fixed period, and exponentially decreasing amplitude. Such a system does not exhibit clustered variation, since its zero-crossings are evenly spaced; in principle, it will cross zero infinitely many times, but only over infinite time. However, if the period depends on an external parameter, and can be made to go to zero by adjusting the parameter, then clustered variation may be achievable. For example, if you run your finger along a vibrating violin string, then the frequency is inversely proportional to the distance from the contact point to the bridge. In principle, if you move your finger toward the bridge in such a way that the distance from your finger to the bridge is proportional to $(t0 - t)^4$, then the string crosses the center infinitely often. Note that the specified value of the exogenous parameter here — the position of the finger — is perfectly well-behaved mathematically.

Note that any single point on the string crosses the center only finitely many times. Thus, the motion of each single point generates only discrete variation; it is the motion of the string as a whole that generates clustered variation. (See section 6 for further discussion.) However, if two fingers are moved together on the string, then the point at the meeting point crosses the center infinitely often.

4 INFINITE SEQUENCES OF EVENTS

In this section, we consider the analogue of clustered variation for events and actions; should it be considered possible for infinitely many events to occur without overlapping during a finite interval?

Certainly, if we admit event types such as "O moves from $P1$ to $P2$," or "O moves distance X in direction D," then Zeno's paradoxes show that the occurrence of infinite sequences of events is unavoidable. The motion of the arrow from 0 to 1 entails the occurrence of the infinite sequence, "Arrow moves from 0 to 1/2; arrow moves from 1/2 to 3/4; ..." Though this looks odd, it is actually not problematic; it is just a clumsy way of speaking about a simple reality. As an analogy, we may observe that, if the sequence of discrete events "$< A; B; C; D; E >$" occurs then concurrent with occurrence of event C are the occurrences of events such as "$< B; C >$," "$< C; D >$", "$< A; B; C >$" ...The fact that nine such events are occurring simultaneously is not a problem, just a clumsy description.

The real problems begin when we switch from descriptive to imperative mode, and talk about plans rather than event descriptions. Do we want to say that Achilles can carry out the plan,

```
procedure P1
while (place(me) ≠ 1)
      move-to((1 + place(me))/2)
```

starting at place=0 by running forward to 1? If so, what about the plans

```
procedure P2
begin D := 1;
      while (true) begin D := D/2;
                         go_forward(D) end end

procedure P3(D)
begin go_forward(D/2); P3(D/2) end

procedure P4(D)
begin P4(D/3); go_forward(D/3); P4(D/3) end
```

("move_to(X)" and "go_forward(D)" are taken to be primitive calls to the effectors.)

Obviously, a plan interpreter implemented along the lines of a conventional programming language interpreter will bomb with each of these; they will either go into an infinite loop that takes infinite time, blow the stack, or run into underflow. However, if plans are viewed as descriptions of actions to be reasoned about rather than as programs to be blindly executed, then it can be argued that these plans should be admitted just as we admitted the event descriptions discussed above.

In fact, there are natural definitions of the semantics of planning languages that admits these plans as correct. For example, most semantics for plans with conditionals and loops (e.g. (McDermott 1982) (Manna and Waldinger 1986)) assume that the test is evaluated instantaneously, and that the body of the conditional begins simultaneously with the conditional. If so, and we idealize real arithmetic as perfect, then P1 is valid. P2 can be made valid if (a) the termination condition on a while loop is interpreted as the condition under which another iteration takes place, rather than a condition that must be false when the loop exits; and (b) the time required for assignment statements is allowed to become arbitrarily small. The recursive procedures P3 and P4 are valid if a suitable minimal fixed-point definition of recursion is used. (In effect, one defines the execution of any infinitely deep calling sequence as completed instantaneously.)

Why should we care what the semantics does, given that no plan interpreter can execute these plans, and no planner will ever come up with them? Actually, these assumptions may not be true. At the interpretive side, it is not hard to imagine a plan optimizer that could recognize one or another of these forms and compile it into the instruction to move forward the whole distance. (Compare "tail-recursion" optimization, which allows an infinite loop to be written recursively, without blowing the stack.) At the planning side, it is not as obvious as it looks at first glance that no planner will ever come up with these. If you apply a GPS-like strategy to Achilles' problem of overtaking the tortoise then the infinite loop

while (place(me) $\neq$ place(tortoise))
    move-to(place(tortoise))

is exactly what you generate. In fact, if the motion of the tortoise is not known in advance and cannot be extrapolated, all plans for overtaking the tortoise will involve something like this. Achilles can overtake a slowly moving butterfly, but he must make finer and finer adjustments as he gets closer and closer. (His motion will not, in general, observe the condition of piecewise analyticity discussed in section 5.) Of course, this is only a truly infinite loop if one assumes infinitely precise perception and motion, but these are natural high-level idealizations of robotic behavior.

5 CONDITIONS ON FLUENTS

In this section, we consider how requiring discrete variation affects the range of states and fluents that can be considered. We consider fluents onto finite ranges (section 5.1) real-valued fluents (5.2); and spatial fluents (5.3). We also show that guaranteeing discrete variation also often involves restricting the local area, or the universe as a whole, to contain only finitely many objects. The primary significance of these results is to provide safety conditions for exogenous parameters in physical theories. If it is necessary to require the endogenous parameters of a system to vary discretely, then the exogenous parameters must be constrained so that this requirement can be fulfilled.

To begin with, we present two equivalent formal definitions of discrete variation.

Definition 1: A non-empty subset I of the real line is an *interval* iff it satisfies the following condition: if $X \in I$, $Y \in I$, and $X < Z < Y$ then $Z \in I$. This includes open, closed, half-open, and single-point intervals, bounded or unbounded.

Definition 2: An interval I is a *zone* of Boolean fluent A if A has the same truth value throughout I. I is a *maximal zone* of A if I is a zone of A and no interval proper containing I is a zone of A.

Definition 3.A: A Boolean fluent A has *discrete variation* if any bounded interval I contains only finitely many maximal zones.

The above definition intuitively corresponds to the desired meaning of clustered variation. However, it uses a high-order expression, posed in terms of the cardinality of a set of intervals, each interval being a set of situations. This may make it difficult for the human reasoner to use, and even more difficult for the automated reasoner.

An alternative equivalent definition, expressible purely in terms of situations,can be formulated:

Definition 3.B: Fluent A has discrete variation if, for each situation S, A has a constant truth value throughout some open interval $(S1, S)$ ending in S and a constant truth value throughout some open interval $(S, S2)$ starting in S. Note that the condition "A has a constant truth value throughout (SP, SQ)" can be expressed purely in terms of situations.

$$\text{constant}(A, SP, SQ) \Leftrightarrow$$
$$[SP < SQ \wedge$$
$$[\forall_{SX,SY}\ SP < SX < SY < SQ \Rightarrow$$
$$[\text{holds}(SX, A) \Leftrightarrow \text{holds}(SY, A)]]].$$

$$\text{discrete}(A) \Leftrightarrow$$

$$[\forall_S \exists_{S1,S2}\ \text{constant}(A, S1, S) \wedge \text{constant}(A, S, S2)]$$

Definition 3.B uses the same construction as axiom 9 of (McDermott 1982). The proof that these two definitions are equivalent is given in (Davis 1991b).

(The following point has created some confusion: The Skolem-Lowenheim theorem implies that "finiteness" is not axiomatizable in a first-order theory. How, then, can definition 3.B, a purely first-order statement, be equivalent to 3.A, a finiteness condition? The answer is that two are equivalent in the standard model of the reals. The first-order axiomatization of the reals also admits non-standard models; in these, 3.B is equivalent to the condition that A has a hyperfinite number of maximal zones in any bounded interval.)

The following useful fact governing Boolean combinations of states is easily shown:

Theorem 4: If $A1 \ldots Ak$ each has discrete variation, then any Boolean combination of them also has discrete variation.

5.1 FLUENTS WITH FINITE RANGE

Let F be a fluent with a finite range D. The state $F(t) = c$ varies discretely for every constant $c \in D$ if and only if F changes values only finitely many times in each finite interval.

Let $F1 \ldots Fk$ be fluents with finite range and discrete variation; let O be a k-place function, and let P be a k-place relation. Then the fluent $\lambda(t)\, O(F1(t) \ldots Fk(t))$ and the state $\lambda(t)\, P(F1(t) \ldots Fk(t))$ vary discretely. Recursively, any fluent or state define as a ground term over $F1 \ldots Fk$ varies discretely.

5.2 REAL-VALUED FLUENTS

Real-valued fluents are trickier. The requirement that $F(t)$ change its value only finitely many times in a finite interval is much too strong; it would rule out any function that changed continuously, such as the identity function $F(t) = t$. On the other hand, the requirement that the state $F(t) = c$ have discrete variation for each constant c is much too weak. The function

$$F(t) = \begin{cases} t & \text{if } t \text{ is rational} \\ -t & \text{if } t \text{ is irrational} \end{cases}$$

has discrete variation for each state $F(t) = c$, but has clustered variation for states $F(t) > c$.

We can rule out such anomalies by requiring that both states $F(t) = c$, and $F(t) > c$ have discrete variation. But even this stronger condition does not allow us to do everything we want to. For example, the function $F(t) = t$ and $G(t) = t + t^3 \sin(1/t)$ both satisfy this condition. (Note that $G(t)$ is monotonically increasing in a neighborhood of $t = 0$.) However, the state $G(t) > F(t)$ has clustered variation in the neighborhood of $t = 0$. We would like to be able to perform basic operations on our fluents without worrying about getting into trouble.

Luckily, there is a helpful theorem about power series. First, two definitions:

Definition 5: A function $f(x)$ on the reals is ***analytic*** at point x_0 if

i. $f(x)$ is continuous and infinitely differentiable at x_0;

ii. The Taylor series

$$f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2!}(x - x_0)^2 f''(x_0) + \ldots$$

converges to $f(x)$ for all x in some neighborhood of x_0.

(Contrary to an error in (McDermott 1982), condition (i) alone does not guarantee discrete variation.)

Definition 6: A function f is analytic over an interval I if f is analytic at every point $x \in I$.

The following theorems are well known:

Theorem 7: If f is analytic at x_0, then the states $f(x) = c$ and $f(x) > c$ vary discretely in a neighborhood of x_0.

Theorem 8: If f and g are analytic, then the functions $f+g$, $f-g$, $f \cdot g$ and $f(g(x))$ are all analytic. The integral and derivative of f are analytic. The quotient $f(x)/g(x)$ is analytic except for x where $g(x) = 0$. The power $f^\alpha(x)$ is analytic for all x if α is a non-negative integer, and analytic for all x where $f(x) > 0$ for all other α.

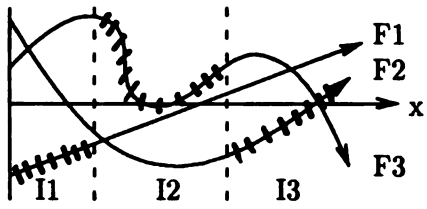
Remark: A lot of good functions are analytic for all x, including polynomials, exponentials, sine and cosine, and so on. The logarithm is analytic for all positive x. Most real-valued functions that come up in practice are analytic for almost all x.

One type of closure that cannot be achieved is closure under solving differential equations. There are differential equations that appear reasonable, but whose solutions include functions with clustered variations. An example is the initial-value system.

$\dot{y} + y^2 = 0.$
$\ddot{x} + y^4 x = 0.$
$y(-1/\pi) = -\pi$; $x(-1/\pi) = 0$; $\dot{x}(-1/\pi) = -\pi$.

For $t < 0$, this has the solution $y(t) = 1/t$; $x(t) = t \sin(1/t)$.

The class of analytic functions, though large, is not adequate for AI applications, which model discontinuous change. Often, the behavior of a fluent will change suddenly as the result of a discrete event such as a switch flipping, a collision of solid objects, a liquid



P(I1)=F1; P(I2)=F3; P(I3)=F2.
The splice s(x) is marked by hatching
Figure 7: Splicing functions

reaching boiling point, and so on. If we posit that such events occur only at discrete times, then we can widen the class of functions to include the corresponding fluents as follows:

Definition 9: Let $\mathcal{F} = \{f_1, f_2 \ldots\}$ be a collection (finite or infinite) of real-valued functions. Let $P(x)$ be a function from the real line to $\mathcal{F}$. Assume that the state $P(x) = f_i$ varies discretely, for each f_i. We then define the *splice* of P as the function $s(x) = (P(x))(x)$ (Figure 7)

Note that P may range over infinitely many different f_i, as long as it changes discretely.

Definition 10: A function $s(x)$ is *piecewise analytic* if there exists a collection $\mathcal{F}$ and a function $P(x)$ such that

i. P satisfies the conditions of definition 9 above.

ii. $s(x)$ is the splice of P.

iii. Each f_i in the image of P is analytic over an open set that contains the closure of $P^{-1}(f_i)$.

Note: In condition (iii) above, f_i must be analytic over the closure of $P^{-1}(f_i)$, not just over $P^{-1}(f_i)$. For example, the function $f(x) = x\sin(1/x)$ is analytic over $(0, \infty)$, but not at 0. Hence, the function defined to be 0 for all $x \leq 0$ and $x\sin(1/x)$ for all $x > 0$ is not considered piecewise analytic.

We then have the following theorems:

Theorem 11: If f is piecewise analytic, then the states $f(x) = c$ and $f(x) > c$ vary discretely.

Definition 12: Function $f(x)$ *is near 0 at* x_0 if either $f(x_0) = 0$, or if the limit of $f(x)$ is 0 as x approaches x_0 either from below or from above.

Theorem 13: Let $f(x)$ and $g(x)$ be piecewise analytic functions. Then $f + g$, $f - g$, $f \cdot g$, $f(g(x))$, $\max(f(x), g(x))$, $\min(f(x), g(x))$, and the integral of f are piecewise analytic. The derivative of f is piecewise analytic, assuming that the derivative is assigned an arbitrary value at points where f is not differentiable. g/f is is piecewise analytic as long as f is nowhere near 0. f^a is analytic as long as f is always positive and nowhere near 0.

The proofs are immediate from theorems 7 and 8, together with definitions 9 and 10.

There are some functions that might be useful that are still not included in this definition, including divergent functions, like $1/x$ or $\sec(x)$, and fractional powers, such as $\sqrt{x}$. If we need these, we can define richer classes of functions. For example, these three functions are included in the class of *piecewise Dirichlet functions*. (See (Davis 1991b) for the definition of this class.) This class has the following properties:

- If $f(x)$ is piecewise Dirichlet then the states $f(x) = c$ and $f(x) > c$ vary discretely.
- All piecewise analytic functions are piecewise Dirichlet.
- Let $f(x)$ and $g(x)$ be piecewise Dirichlet functions. Then $f + g$, $f - g$, $f \cdot g$, $\max(f(x), g(x))$, and $\min(f(x), g(x))$, are piecewise Dirichlet. The derivative of f is piecewise Dirichlet, assuming that the derivative is assigned an arbitrary value at points where f is not differentiable. g/f is piecewise Dirichlet as long as f is nowhere equal to zero. f^a is piecewise Dirichlet as long as f is always positive. (Note that, unlike theorem 12, for division and exponentiation f is allowed to converge to zero as long as it does not attain zero.)

However, the Dirichlet functions are not closed under integration or composition.

By theorem 4, if the states $f(x) = c$ and $f(x) > c$ vary discretely, then any finite Boolean combination of these also varies discretely. But a finite Boolean combination of inequalities is just a finite union of intervals. Thus, if S is the union of finitely many intervals, then the state $f(x) \in S$ varies discretely if $f(x)$ is piecewise analytic or piecewise Dirichlet.

What about more general states involving the value of $f(x)$? Clearly, if a state $P(y)$ has clustered variation as a function of y, then it would be unreasonable to expect $P(f(t))$ to vary discretely as a function of t; even such a well-behaved function as $f(t) = t$, would violate that condition. For example, we cannot expect the state, "$f(t)$ is a rational number" to vary discretely. However, it is easily shown that any set of real numbers that does not have clustered variation is the union of a discretely varying collection of intervals. We have already taken care of the case of a finite collection of intervals; the only remaining case, therefore, is an infinite collection of discretely varying intervals, such as "$f(t)$ is an integer," or "$\sin(f(t)) > 0$."

Let I be the union of infinitely many intervals with discrete variation, and let $f(x)$ be a function such that $f(x) > c$ varies discretely for each c. If $f(x)$ is

bounded over every bounded interval, then the state $f(x) \in I$ varies discretely. (Note: a piecewise analytic function must satisfy this condition; a piecewise Dirichlet function need not.) If $f(x)$ diverges for finite x, then, in general, the state $f(x) \in I$ does not vary discretely. Thus, a state such as "$f(t)$ is an integer" or "$\sin(f(t)) > 0$" has discrete variation for functions such as $f(t) = t$ or $f(t) = e^t$, but not for $f(t) = 1/t$.

We run into problems, though, if we have two scales, one of which is the reciprocal of the other. In that case, an infinite sequence of discrete intervals in the one corresponds to a cluster point in the other. For example, consider the relation on tones, "This tone is a C." If we consider the series of higher and higher C-tones in octaves, the frequencies form a discrete series of numbers, but the wavelengths are clustered around zero; if we consider lower and lower C-tones, the wavelengths are discrete but the frequencies are clustered around zero. (Going up an octave doubles the frequency and halves the wavelength.)

It is true that both are discretely spaced on a logarithmic scale. However, a logarithmic scale can only be used if zero and negative values are guaranteed to be meaningless. We can get away with this if we are using the scale only for wavelengths, but not if we wish to use the same scale for non-positive quantities of the same dimensionality, such as distances or coordinates,

Of course, whether the state $Q(f(t))$ has discrete or clustered variation over time does not depend on of the scale used to measure the range space of f. (For example, whether the state "The string is vibrating at the tone of C" varies discretely over a time period is a physical fact, which does not depend on whether we define C in terms of frequency or wavelength.) The point is that it is not sufficient to require that the real-valued fluents be well-behaved; it is also necessary to show that the significant states in their range spaces vary discretely; and this latter assumption is not one that can be made universally.

5.3 SPATIAL FLUENTS

Reasoning about shape and motion raises problems that are analogous to those of real-valued fluents, but in a more complex setting. We wish to guarantee that the times in which a moving point object are inside a specific region vary discretely; that the times in which two moving regions overlap, or abut, or are within a given distance of one another vary discretely; and so on. Ensuring these conditions requires restricting both possible shapes and possible motions.

As in the one-dimensional case, we wish both to include both a wide range of possible shapes and motions, and to attain useful closure conditions. However, I have not been able to prove discrete variation, even over the set of shapes and motions generated by simple kinematic theories, if one wants to include such operations as the region "swept out" by moving region **R** along motion M. The best results that I have been able to prove are given below.

Definition 14: A *region* is a subset of $\Re^k$. Thus, the set of all k-dimensional regions is $\mathcal{P}(\Re^k)$, where $\mathcal{P}$ is the power-set operator.

Definition 15: An *algebraic constraint* is a constraint of the form $f(\vec{x}) > 0$ or $f(\vec{x}) \geq 0$ where f is an algebraic (multinomial) function. A *semi-algebraic* region is the solution space of a finite Boolean combination of algebraic constraints.

Theorem 16: Let **R** be a semi-algebraic region, and let $\vec{x}(t)$ be a piecewise analytic function of time. Then the Boolean fluent $\vec{x}(t) \in \mathbf{R}$ varies discretely.

Proof: Let **R** be defined by the Boolean combination of the functions $f_1, f_2 \ldots f_k$. The functions $f_i(\vec{x}(t))$ are piecewise analytic functions in t; hence each constraint $f_i(\vec{x}(t)) \geq 0$ varies discretely, by theorem 11. Hence, by theorem 4, the Boolean combinations of these states also varies discretely.□

Definition 17: The *algebraic language* of points and k-dimensional regions is the first order language over $\Re \cup \mathcal{P}(\Re^k)$ containing the non-logical primitives '0', '1', 'plus(X,Y)', 'times(X,Y)', 'greater(X,Y)', and 'in$(X_1 \ldots X_k, \mathbf{R})$'. This last primitive is a predicate stating that the point with coordinates $X_1 \ldots X_k$ is an element of region **R**. The other symbols are defined for real-valued arguments and have their usual meaning.

Definition 18: An *algebraic property* of regions $\mathbf{R_1} \ldots \mathbf{R_m}$ is a formula in the algebraic language of points and k-dimensional regions with open variables $\mathbf{R_1} \ldots \mathbf{R_k}$.

Examples:

i. The property "The minimum distance from **R** to **S** is at least D," is algebraic, since it can be written (in two dimensions) as
$$\forall_{<W,X>\in\mathbf{R}} \forall_{<Y,Z>\in\mathbf{S}} (W-Y)\cdot(W-Y)+(X-Z)\cdot(X-Z) \geq D\cdot D$$
(The subtractions above can easily be expressed in terms of plus.)

ii. The property "The volume of **R** is greater than 3," is not algebraic. The definition of volume involves an integral, which is not an algebraic operation.

Theorem 19: Consider a finite collection of objects moving rigidly through space. If the shapes of the objects are semi-algebraic, and the motions are piecewise analytic, then any state corresponding to an algebraic property of the regions they occupy varies discretely.

Theorem 19 is proven by combining theorem 16 above with the classic result of Tarski (1951) that any algebraic property is equivalent to a collection of algebraic

constraints. Details are given in (Davis 1991b).

Definition 20: An *analytic constraint* is a constraint of the form $f(\vec{x}) > 0$ or $f(\vec{x}) \geq 0$ where f is an analytic function. A *semi-analytic* region is the solution space of a finite Boolean combination of analytic constraints.

Theorem 21: Let R be a semi-analytic region, and let $\vec{x}(t)$ be a piecewise analytic function of time. Then the Boolean fluent $\vec{x}(t) \in \mathrm{R}$ varies discretely.

Proof: Identical to proof of theorem 16.

Conjecture 22: Consider two objects that are moving rigidly through space. If the shapes of the objects are bounded and semi-analytic and the motions are piecewise analytic, then any state corresponding to an algebraic property of the regions they occupy varies discretely.

Another issue concerns real-valued functions defined over space, such as temperature or pressure. We would like to guarantee that the values of such functions encountered by an object moving through space are well-behaved. We can achieve this with a generalization of the notion of a piecewise analytic function:

Definition 23: Let $s(\vec{x})$ and $f_1(\vec{x}), f_2(\vec{x}) \ldots$ be functions from $\Re^k$ to $\Re$. Let P be a function mapping each of the functions f_i to a region in $\Re^k$ such that

a. P is disjoint and exhaustive. That is, for each $\vec{x} \in \Re$ there is exactly one f_i such that $\vec{x} \in P(f_i)$.
b. For all i, $P(f_i)$ is a semi-analytic region of $\Re^k$.
c. Any bounded $S \subset \Re^k$ interesects $P(f_i)$ for only finitely many i.
d. f_i is analytic over an open set that includes the closure of $P(f_i)$.
e. For $\vec{x} \in P(f_i)$, $s(\vec{x}) = f_i(\vec{x})$.

Then $s(\vec{x})$ is said to be *piecewise analytic.*

Theorem 24: Let $\vec{x}(t)$ be a piecewise analytic function from time to R^k, and let $s(\vec{x})$ be a piecewise analytic function from $\Re^k$ to $\Re$. Then $s(\vec{x}(t))$ is a piecewise analytic function of time.

Proof: Immediate from the definition and theorem 21.□

5.4 FINITELY MANY OBJECTS

If the universe contains infinitely many objects, then states of the form "There exists an object with property P" may have clustered variation even if each individual object is well-behaved. For example, suppose there is an infinite collection of light bulbs, and bulb #1 is on from $t = 0$ to $t = 1/2$, bulb #2 is on from $t = 3/4$ to $t = 7/8$; bulb #3 is on from $t = 15/16$ to $t = 31/32$ and so on. Then the state "Some light bulb is on" has clustered variation, even though each individual light bulb is well-behaved.

Often, states of physical significance depend only on objects that are in some bounded region. In that case, it suffices to demand that there are only finitely many objects in any bounded region of space-time. Suppose, modifying the above example, we have a rule that a room is illuminated just if it contains a lighted light bulb. In that case, the state "The room is illuminated" can be guaranteed to vary discretely by positing that the state "Light bulb N is on" varies discretely for each N, and that during any finite time interval there are only finitely many light bulbs in the room.

Note that this condition is strictly stronger than the condition, "At any instant, there are only finitely many objects in any spatially bounded region," which does not suffice to guarantee the desired result. Consider the following example. Syldavia is interested in the state "There are no enemy aircraft in Syldavian airspace." Freedonia has an infinite fleet of airplanes, regularly spaced on an infinite landing strip. At time $t = 0$, plane 1 flies from its spot on the landing strip to Syldavia, returning to its spot by $t = 1/2$. Plane 2 does the same from $t = 3/4$ to $t = 7/8$, and so on. No individual plane is doing anything extraordinary (except for going faster than the speed of light); nonetheless, the state "There are no aircraft over Syldavia" has clustered variation.

The condition of only finitely many objects in a bounded region of space-time may, in turn, require restrictions on processes that govern the behavior of the objects. For instance, if we replace the airplanes of the previous example by bullets fired from guns spread out on the landing strip, with further guns firing faster bullets, then we must posit that the guns do not fire in such a way as to send infinitely many bullets through a finite space in finite time. This, note, restricts the process allowed to happen over regions unbounded in both space and time. (If gun I is at $x = 2^{4I}$ and fires a bullet in the negative direction at time $-2^{2I} - 1/2^{2I-1}$ at speed 2^{2I}, then the bullet will be between $x = 0$ and $x = -1$ between the times $-1/2^{2I-1}$ and $-1/2^{2I}$.) Of course, the guns here are superfluous; one can imagine that the bullets have been pursuing these trajectories since time immemorial, never looking in the least clustered until $t = 0$. All in all, it may simply be best to posit that in all space-time there are only finitely many objects.

Another example: suppose that the bouncing ball of section 1 is slightly brittle, and that each time the ball hits the ground a small piece breaks off whose size is proportional to the energy of the collision. Then, after the bouncing ball has attained a state of rest, the floor will be littered with an infinite collection of diminishing objects, which can proceed to make trouble.

It is often useful to require that only finitely objects exist in an bounded spatial region for reasons independent of temporal considerations. For instance, in the blocks world, one would like to infer that, if a block is

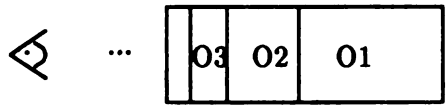
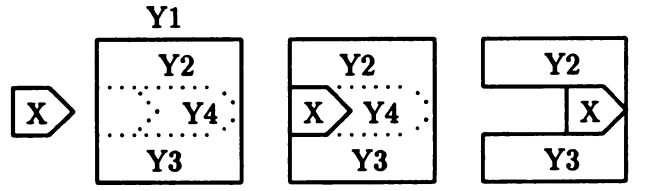


Figure 8: Infinitely Many Occluding Objects

on top of a tower of blocks, then it is resting on some particular other block; this is guaranteed if there are only finitely many blocks in the tower. Similarly, in a theory of perception, one would like to infer that, if there is an object blocking your view in a particular direction, then there is a frontmost object in that direction; again, this is guaranteed if there are only finitely many objects in the neighborhood, and objects are reasonably shaped. (Figure 8)



As blade X cuts through target Y, it first annihilates the overall chunk Y1. Later, it annihilates Y4 as well, leaving Y2 and Y3.

Figure 9: Cutting in terms of chunks

6 ONTOLOGY

What kinds of scenarios are considered examples of clustered variation and what kinds of formal difficulties are involved in such scenarios may depend critically way in which the universe is divided into entities.

One example was mentioned in section 3. Suppose I have a vibrating violin string, and I run my finger to the bridge in a finite time in such a way that the part of the string between the finger and the bridge vibrates infinitely often. If we consider "The vibrating part of the string" to be an entity, then the state "The vibrating part of the string is to the left of center" has clustered variation. However, if we take the individual points on the string as primitive entities, then these are all well behaved. Each point crosses the center line a finite number of times, and then, being on the far side of the finger, remains still.

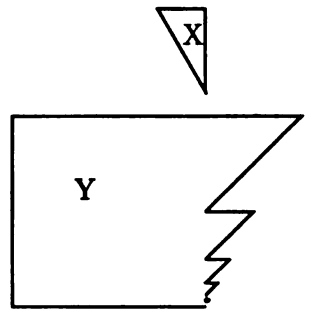
In a recent paper axiomatizing the process of cutting solid objects (Davis 1991a), I ran into a complex form of this issue. The paper discusses an idealized model of cutting in which the blade changes the shape of the target by simply annihilating the material that it sweep through. Two alternative ontologies for this model are presented. The first ontology views the target as an object whose shape is gradually modified by the blade until it is split, at which time the original object ceases to exists and two new objects come into being. In this theory, there are three central axioms of change:

1. If an object exists from $S1$ to $S2$, then its shape in $S2$ is equal to its shape in $S1$ minus whatever has been carved out of it.
2. Object O ceases to exist in situation S if and only if its shape is connected up until S and becomes disconnected in S.
3. Object O comes into existence in situation S if and only if it is one connected component of another object $O2$ that ceases to exist in S.

These axioms serve both as causal axioms and as frame axioms. The first axiom is causal if something has been carved out of the object, and is a frame axiom if nothing has. The second and third axioms are causal in the direction "Change if condition" and frame axioms in the direction "Change only if condition."

The second ontology takes a chunk of stuff as its primary type of entity. There is one chunk for every connected subset of the material of an object; thus, there are uncountably many chunks at any instant. A chunk has a fixed shape. It ceases to exist when any part of its material is penetrated by a blade. The one rule of change in this theory is that a chunk is annihilated if and only if it is penetrated by a blade. (Figure 9).

The first ontology is extremely sensitive to clustered variation. Since dramatic discrete changes occur whenever an object is split, the prospect of infinitely splits occurring in finite time sends the whole theory into conniptions. In a situation like figure 10, where, intuitively, the blade cuts an infinite sequence of small objects on the right off the large block at the left, the theory does not permit the block on the left to exist when the blade has reached the bottom. An object, in this theory, can come into existence only be being split off a previous object; and here there is no previous objects, only a blur of previous objects. (Keep in mind that the object changes identity each time a split is complete.) It is therefore necessary, in this theory, to enforce rigidly the condition of discrete variation; specifically, we require that any bounded region of space-time can contain only finitely many objects. (It is an open problem to find reasonable restrictions on motions and shapes sufficient to guarantee this con-



The piece of Y on the left changes identity infinitely often as X cuts through.

Figure 10: Anomaly in object theory.

dition.)

By contrast, these anomalies give no trouble to chunk theory. In chunk theory, splitting looks exactly the same as shape deformation; history consists purely of a stream of chunks being destroyed. The predictions of the theory are well-defined and reasonable for any specification of initial shapes and motions that satisfy basic topological constraints.

The two theories are provably equivalent in cases that obey the restriction that any bounded region of space-time contains only finitely many objects.

7 CONCLUSION

Perhaps the whole issue still seems spurious and irritating. However, if we follow the conventional practice in knowledge representation of characterizing theories rigorously, and the conventional practice in physical reasoning of using real-valued time and space and using simple physical approximations, then these problems are unavoidable. Those researchers who have tried to develop formal theories of physical reasoning based on real-valued quantities will, I think, agree that these infinitary paradoxes are not a side issue, of interest only to those with a prurient taste for paradox. Rather, developing such theories is like designing structures for the area around the San Andreas Fault. Making a structure earthquake-proof is an irritating distraction from the main task of building a useful structure, but it is really not safe to ignore the issue. Nor is there a simple formula that can be uniformly applied; there are some general techniques, but each new type of structure must be thought through individually.

Acknowledgements

This research has been supported by NSF grant #IRI-9001447. Thanks to Drew McDermott for getting me started on this problem, and to my father Philip Davis and Bud Mishra for some mathematical pointers.

References

Davis, E. (1989a). "Solutions to a Paradox of Perception with Limited Acuity," in R. Brachman, H. Levesque, and R. Reiter (eds.) *First International Conference on Knowledge Representation and Reasoning,* Morgan Kaufmann, 1989.

Davis, E. (1989b). "Order of Magnitude Reasoning in Qualitative Differential Equations," in J. de Kleer and D. Weld (eds.), *Readings in Qualitative Physical Reasoning,* Morgan Kaufmann, pp. 422-434.

Davis, E. (1991a). "The Kinematics of Cutting Solid Objects," Tech. Rep. 541, NYU Comp. Sci. Dept.

Davis, E. (1991b). "Infinite Loops in Finite Time: Some Observations," Tech. Rep. 599, NYU Comp. Sci. Dept.

Fleck, M. (1988). "Boundaries and Topological Algorithms," M.I.T. AI Lab, Tech. Rep. 1065.

Forbus, K. (1985). "Qualitative Process Theory," in D. Bobrow (ed.) *Qualitative Reasoning about Physical Systems.* MIT Press.

Galton, Anthony (ed.) (1987). *Temporal Logics and Their Applications,* Academic Press.

Hamblin, C.L. (1971). "Instants and Intervals," *Studium Generale,* vol. 24, pp. 127-134.

Hayes, P. (1985). "The Second Naive Physics Manifesto," in J. Hobbs and R. Moore (eds.) *Formal Theories of the Commonsense World,* Ablex Pubs.

Kaufmann, S. (1991). "A Formal Theory of Spatial Reasoning," in J. Allen, R. Fikes, and E. Sandewall (eds.) *Proc. Second Intl. Conf. on Princ. of Knowledge Representation and Reasoning,* Morgan Kaufmann

Manna, Z. and R. Waldinger, (1986). "A Theory of Plans," in M. Georgeff and A. Lansky (eds.) *Reasoning about Actions and Plans,* Morgan Kaufmann.

McDermott, D. (1982). "A Temporal Logic for Reasoning about Processes and Plans," *Cognitive Science,* vol. 6, pp. 101-155.

Russell, B. (1919). *Introduction to Mathematical Philosophy,* George Allen and Unwin, Ltd., London.

Tarski, A. (1951). *A Decision Method for Elementary Algebra and Geometry,* University of California Press.

Reasoning About Indefinite Actions

L. Thorne McCarty
Ron van der Meyden*
Computer Science Department
Rutgers University
New Brunswick, NJ 08903

Abstract

In this paper, we view planning as a special case of reasoning about indefinite actions. We treat actions as predicates defined over a linear temporal order. This formalism permits the representation of concurrent activity. Suppose we have a set of abstract actions defined by Horn clauses from a set of basic actions. Let us assume that an abstract action ψ has occurred, and ask whether a given condition ϕ is entailed by all the basic actions that constitute ψ. A countermodel to this hypothetical implication is then a plan for "doing ψ and avoiding ϕ." We propose a formalization of this problem using circumscription, and argue that this is the correct formalization if our action definitions include recursive rules. We then investigate two techniques for solving the problem: (1) a special type of inductive proof procedure, which is sound but not complete; and (2) a decision procedure that works for an interesting subclass of the general problem. Because of the use of recursive rules in our action definitions, we can use these techniques to reason about an abstract action that consists of "doing ψ some finite number of times." We demonstrate the techniques on a simple example involving concurrent, repetitive actions.

1 Introduction

Planning problems often take the following form: There is an abstract action that we wish to perform, and a number of concrete ways to perform it. Some of these concrete actions have negative consequences. Is there some way to carry out the abstract action such that the negative consequences do not occur?

For example, consider the situation facing Evelyn F. Gregory in 1928. She was the owner of all the outstanding shares of the United Mortgage Corporation, which in turn owned 1000 shares of the Monitor Securities Corporation. The Monitor shares had appreciated substantially in value, and Mrs. Gregory wanted the United Mortgage Corporation to distribute them to her as a dividend. But this transaction, if carried out directly, would be taxable at ordinary income rates. Was there some way to structure the transaction to avoid this result? Mrs. Gregory's lawyer had an answer. A new company, named Averill, was created and incorporated in Delaware, and the Monitor shares were transferred to the Averill Corporation, in exchange for which Averill issued all of its stock directly to Mrs. Gregory. According to the Revenue Act of 1928, Section 112(i), these events were not taxable. Four days later, the Averill Corporation was liquidated, and the Monitor shares were distributed to Mrs. Gregory, its sole stockholder. This distribution, although taxable, would be treated as a capital exchange. Mrs. Gregory's lawyer had thus succeeded in converting an ordinary dividend into a capital gain, which was taxable at a much lower rate. The Board of Tax Appeals agreed with this characterization of the situation. *Evelyn F. Gregory*, 27 *B.T.A.* 223 (1932).

In this paper, we will investigate a general approach to planning problems of this sort. We assume that we have a number of *abstract actions* which are built up by *definitions* from a set of *basic actions*. These actions, both abstract and basic, will be interpreted over a *linear temporal order*. For example, 'DistributeAssets(umc, efg, t_1, t_2)', with $t_1 < t_2$, might be just such an abstract action. In addition, we will define various *queries* over this linear temporal order. For example, 'Tax(efg, t_2)' might be just such a defined query. Now consider the following implication:

$$\text{Tax}(\text{efg}, t_2) \Leftarrow \text{DistributeAssets}(\text{umc}, \text{efg}, t_1, t_2) \quad (1)$$

We interpret this as a *goal* to be *proven*. It means:

*The final version of the paper was prepared while the second author was visiting the School of Computer Science and Engineering, University of New South Wales, Kensington NSW, Australia.

"If we assume that 'DistributeAssets(umc, efg)' occurs over the interval from t_1 to t_2, does it follow that 'Tax(efg)' is true at t_2?" If the answer is 'Yes', then we know that every possible combination of basic actions that constitutes the defined action 'DistributeAssets(umc, efg, t_1, t_2)' results in a situation in which 'Tax(efg, t_2)' is true. Presumably, this is the conclusion that the Internal Revenue Service would like to establish. If the answer is 'No', then we know that there is some combination of basic actions that satisfies the definition of the abstract action 'DistributeAssets(umc, efg, t_1, t_2)' in which 'Tax(efg, t_2)' is not true. In other words, there exists a *countermodel* to the hypothetical implication in (1). This is the conclusion that the taxpayer would like to establish.

We thus view planning as a special case of reasoning about indefinite actions, a position with numerous antecedents in the AI literature. The first explicit statement of the close relationship between "Planning and Acting" was Drew McDermott's 1978 article with that title. "[P]roblem solving is part of the study of *action*," McDermott wrote [1978, p. 72]. "A problem is a difficult action." Within this tradition, various logics have been proposed for the representation of actions [McDermott, 1982; Allen, 1984; Georgeff and Lansky, 1985; Kowalski and Sergot, 1986; Shoham, 1987; Manna and Waldinger, 1987].

By formalizing actions as predicates over a linear temporal order and allowing abstract actions to be built up by means of definitions, we are able to give *hierarchical* definitions of *concurrent* actions, and can represent *non-linear plans* by giving incomplete information about the linear order. In this respect our work is closely related to an early paper by Allen and Koomen [1983] and a more recent paper by Eshghi [1988]. Allen and Koomen represent an action by a conjunction of predicates that hold over temporal intervals and can be decomposed, in turn, into more primitive predicates over temporal intervals. Eshghi's representation, using the Event Calculus of Kowalski and Sergot [1986], has a similar expressive power. In addition, Eshghi proposes a form of reasoning based on *abduction* [Peirce, 1931] rather than deduction. Whereas Allen and Koomen write the expansion of an abstract action as a *necessary* condition, Eshghi writes it initially as a *sufficient* condition and then turns the implication around by an abductive inference. This abductive approach to planning has subsequently been followed up by others [Shanahan, 1989; Missiaen, 1991; Denecker *et al.*, 1992].

Although our ultimate goal is to reason about actions with the complexity apparent in legal domains (see [McCarty, 1989; Schlobohm and McCarty, 1989]), our objectives in the present paper are more modest. We will study hypothetical implications in which the antecedent – e.g., 'DistributeAssets(umc, efg, t_1, t_2)' in (1) – is an action defined by a set of Horn clauses. The work of Allen and Koomen [1983] and Eshghi [1988] essentially allows the definition of actions by a set of *nonrecursive* Horn clauses. We will extend this work by allowing *recursive* definitions as well. There are many situations in which recursive definitions are needed, one of the most important being the representation of repetitive actions. Given any action '$A(t_1, t_2)$', we often want to talk about an action '$R(t_1, t_n)$' which consists of "doing A some finite number of times." For example: "George ran around the track some finite number of times." "The United Mortgage Corporation created some finite number of subsidiaries." "The Monitor shares were transferred to Mrs. Gregory through some finite chain of intermediaries." We can obviously represent these actions if we allow recursive definitions in our language. However, if we wish to use a recursively defined action in the antecedent of a hypothetical implication, as in (1), it is necessary to go beyond first-order logic, as we will see in Section 2 of this paper.

Section 2, following our work in [McCarty and Meyden, 1991], suggests that the proper way to formalize this problem is to *circumscribe* [McCarthy, 1980; Lifschitz, 1985] the action definitions. We give a precise characterization of the *circumscriptive query problem*, and illustrate it with a simple example of an inference involving concurrent, repetitive actions. However, circumscription is a second-order formalism. Can we ever hope to compute the correct response to a circumscriptive query? The answer is: Yes. The main technical contribution of the present paper is the development of two techniques for deciding whether a hypothetical implication is entailed by the circumscription of a set of action definitions. The first technique, presented in Section 3, is based on a special type of proof by induction, and is therefore sound but not complete. The second technique, presented in Section 4, is an actual decision procedure, which works for a restricted, but useful, class of queries. We illustrate the techniques in these two sections by solving the simple problem introduced in Section 2.

In Section 5, we present another example which shows how a planning problem can be solved by the construction of a countermodel to a hypothetical implication. This example also shows how the "closed world assumption" can be incorporated into our formalism. Section 6 then compares our approach to related work.

2 Circumscribed Action Definitions

In a recent paper [McCarty and Meyden, 1991], we suggested that indefinite information arises in common sense reasoning from the *circumscription* [McCarthy, 1980; Lifschitz, 1985] of definite rules. We will review our analysis briefly here, and then show how to extend it to the case of indefinite actions.

We begin with a Gedanken experiment. Suppose we have a set of *basic predicates*: 'Block(x)', 'Red(x)', 'Green(x)', 'On(x, y)', that express observable facts about the world. Suppose we also have sufficient conditions for a set of *defined predicates*, as follows:[1]

$$\textbf{ChristmasBlock}(x) \Leftarrow \text{Block}(x) \wedge \text{Red}(x) \quad (2)$$

$$\textbf{ChristmasBlock}(x) \Leftarrow \text{Block}(x) \wedge \text{Green}(x) \quad (3)$$

$$\textbf{Above}(x, y) \Leftarrow \text{On}(x, y) \quad (4)$$

$$\textbf{Above}(x, y) \Leftarrow \text{On}(x, z) \wedge \textbf{Above}(z, y) \quad (5)$$

Now imagine a two-person communication situation in which the "speaker" applies these definitions to a world of observable facts, and reports some of his or her conclusions. For example, the speaker might report: '**ChristmasBlock**(a)' and '**Above**(a,b)'. It is our job (as the "hearer") to make inferences about the actual state of the world, even though we have not observed it directly. Note that this is essentially an *abductive* task [Peirce, 1931].

How should we formalize this problem? For the non-recursive rules (2)–(3), Clark's Predicate Completion [Clark, 1978] gives us the correct results. For example, the following implication is entailed by the completion of (2) and (3):

$$\text{Red(a)} \vee \text{Green(a)} \Leftarrow \textbf{ChristmasBlock}(a),$$

and we can therefore conclude that 'a' is either 'Red' or 'Green'. However, for the recursive rules (4)–(5) the situation is more complex. Consider the following hypothetical implication:

$$(\exists w)\text{On}(w, \text{b}) \Leftarrow \textbf{Above}(a, b). \quad (6)$$

Intuitively, if we have been told that '**Above**(a,b)' is true according to the definition in (4)–(5), then we ought to conclude that there is something on 'b'. But the completion of (4)–(5) would give us:

$$\textbf{Above}(x, y) \Rightarrow \text{On}(x, y) \vee (\exists z)[\text{On}(x, z) \wedge \textbf{Above}(z, y)], \quad (7)$$

and (4)–(5) and (7) together do not entail (6). The solution to this problem is to use the *circumscription* of '**Above**' in rules (4)–(5), instead of using predicate completion. We refer the reader to [McCarty and Meyden, 1991] for the details.

The extension of these ideas to action definitions is straightforward. Suppose we have a set of *basic actions*: 'A(x, t_1, t_2)', 'B(x, y, t_1, t_2)', 'C(y, t_2)', in which x and y are *object variables* and t_1 and t_2 are *order variables*. Intuitively, 'A(x, t_1, t_2)' asserts that some observable fact about x is true over the time interval from t_1 to t_2. We now add *abstract actions*: '**P**(x, t_1, t_2)', '**R**(x, y, t_1, t_2)', etc. Abstract actions are defined by Horn clauses that are allowed to contain *order relations* in their antecedents, but may not contain function symbols. For example, we could define '**P**(x, t_1, t_2)' as follows:

$$\textbf{P}(x, t_1, t_2) \Leftarrow \text{A}(x, t_1, t_2) \wedge t_1 < t_2 \quad (8)$$

$$\textbf{P}(x, t_1, t_2) \Leftarrow \text{A}(x, t_1, t_3) \wedge \textbf{P}(x, t_3, t_2) \wedge t_1 < t_3 < t_2 \quad (9)$$

The only difference between these action definitions and the definitions studied in [McCarty and Meyden, 1991] is the special treatment of order variables and order relations, but this difference is quite significant. The relation '$t_1 < t_2$' is interpreted over a *linear order*, and it means: "t_1 is strictly less than t_2." In other words, '<' is both transitive and irreflexive, and it satisfies the disjunctive constraint: $t_1 < t_2 \vee t_1 = t_2 \vee t_2 < t_1$. Because of this constraint, the order relations provide an additional source of indefinite information in our language.

As illustrated in (8)–(9), action definitions can be recursive. Thus the argument in [McCarty and Meyden, 1991] that Clark's Predicate Completion should not be used as a formalization of our Gedanken experiment applies here as well. Instead, we will *circumscribe* the defined actions. Let $\mathcal{R}$ be the set of action definitions, and let $P = <P_1, P_2, \ldots, P_k>$ be a tuple consisting of all the predicates that appear on the left-hand side of the rules in $\mathcal{R}$. We denote the circumscription of P in $\mathcal{R}(P)$ by $Circ(\mathcal{R}(P);P)$. When we wish to talk about predicate completion, we write $Comp(\mathcal{R})$ instead. Now let ϕ, the query, be a *positive existential formula* constructed from basic actions, abstract actions, and (possibly) order relations. If ϕ contains no abstract actions, it will be called a *basic* query. Let ψ be a *conjunction* of basic actions, abstract actions, and (possibly) order relations. If there are free variables in ϕ or ψ, we assume that the goal $\phi \Leftarrow \psi$ is universally quantified at top level. The *circumscriptive query problem* is now: $Circ(\mathcal{R}(P); P) \models \phi \Leftarrow \psi$.

Although the hypothetical implication (1) in Section 1 satisfies our constraints on ϕ and ψ, the action 'DistributeAssets(umc, efg, t_1, t_2)' is too complex to analyze here. We will therefore consider a simpler example, suggested by the following story:

Mermaids Get the Sack

Ethel and Daphne Mermaid are synchronized swimmers in the employ of Sam Silverscreen, movie mogul. On the set of "Swimming in the Rain," Sam is giving Ethel and Daphne instructions for the next scene. "What I want you to

[1] For clarity, we use boldface type for the defined predicates. We also use italic type for variables and roman type for individual constants.

do today is very simple," Sam says. "Just keep swimming laps of the pool. As soon as you finish a lap, start the next. But, whenever you both start a lap at the same time you must also finish it at the same time. Don't stop till I tell you to."

"OK, boss," reply Ethel and Daphne (even their speech is synchronized.)

"Lights! Camera! Action!" says Sam, and the Mermaids start their synchronized swimming. Sam retires to his office for a nap on the casting couch.

When Sam returns some time later, he notices that Ethel has just completed a lap, whereas Daphne is still half way through hers. "Cut! Cut!" he yells when Daphne reaches the end of the pool. "Did either of you stop swimming at any stage?" he asks.

"No, boss," reply the Mermaids.

"Well, you're both fired anyway," says Sam.

Question: Why did the Mermaids get the sack?

Here is a formalization of the story: Let $\mathbf{A}(x, t_1, t_2)$ represent the action in which Mermaid x swims one lap of the pool over the time interval from t_1 to t_2. The situation that would lead to the Mermaids' dismissal, assuming that neither stopped swimming, is then:

$$(\exists x, y, v_1, v_2, v_3) \quad [\mathbf{A}(x, v_1, v_2) \wedge \mathbf{A}(y, v_1, v_3) \wedge v_1 < v_2 < v_3] \tag{10}$$

Call this ϕ. Sam Silverscreen knows that each Mermaid has completed some finite number of laps, and we can represent this situation recursively. Define the abstract action '$\mathbf{P}(x, t_1, t_2)$' by the Horn clauses in (8)–(9). Then '$\mathbf{P}(\mathrm{d}, u_1, u_2)$' is the action in which Daphne swims some finite number of laps from u_1 to u_2, and '$\mathbf{P}(\mathrm{e}, u_1, u_3)$' is the action in which Ethel swims some finite number of laps from u_1 to u_3. Figure 1 shows the situation that Sam observes (and infers) when he returns from his nap at time u_3. Thus we need to establish:

$$\phi \Leftarrow \mathbf{P}(\mathrm{d}, u_1, u_2) \wedge \mathbf{A}(\mathrm{d}, u_2, u_4) \wedge \mathbf{P}(\mathrm{e}, u_1, u_3) \wedge u_1 < u_2 < u_3 < u_4 \tag{11}$$

If we can show that (11) follows from the circumscription of '$\mathbf{P}$' in (8)–(9), then we know with certainty that the Mermaids disobeyed their instructions and were (legally!) fired.

How can we show this? First, let us recall some results from [McCarty and Meyden, 1991; Meyden, 1992 in press], which dealt with the simpler framework in which there are no order relations:

Theorem 2.1 *For arbitrary $\mathcal{R}$, ϕ, ψ not containing order relations, the circumscriptive query problem $Circ(\mathcal{R}(P);P) \models \phi \Leftarrow \psi$ is not decidable. However, the set $\{\langle \mathcal{R}, \phi, \psi \rangle \mid Circ(\mathcal{R}(P);P) \not\models \phi \Leftarrow \psi\}$ is recursively enumerable.*

Thus, since adding order relations can only add to the complexity of the circumscriptive query problem, we cannot hope to have decidability in general, nor even a complete proof theory, since we are dealing with an undecidable set whose complement is recursively enumerable. However, we also identified in [McCarty and Meyden, 1991; Meyden, 1992 in press] a decidable subproblem:

Theorem 2.2 *For arbitrary $\mathcal{R}$, ψ and* basic *queries ϕ, none of which contain order relations, the circumscriptive query problem $Circ(\mathcal{R}(P);P) \models \phi \Leftarrow \psi$ is decidable.*

This gives us some hope that a similar result may hold when we add order relations. This would suffice for a solution of the Mermaid problem, since the query ϕ for this problem is basic. Unfortunately, the generalization does not go through.

Theorem 2.3 *For arbitrary $\mathcal{R}$, ψ and basic queries ϕ containing order relations, the circumscriptive query problem $Circ(\mathcal{R}(P);P) \models \phi \Leftarrow \psi$ is not decidable.*

Thus, adding order relations makes the circumscriptive query problem strictly more complex. However, the complementary problem has not increased in complexity:

Theorem 2.4 *For arbitrary $\mathcal{R}$, ψ, ϕ containing order relations, the set $\{\langle \mathcal{R}, \phi, \psi \rangle \mid Circ(\mathcal{R}(P);P) \not\models \phi \Leftarrow \psi\}$ is recursively enumerable.*

Theorem 2.4 shows that the planning problem, as we defined it in Section 1, is semidecidable: If there exists a countermodel to the hypothetical implication $\phi \Leftarrow \psi$, then we will eventually be able to find it. But Theorem 2.3 and Theorem 2.4 together show that even for basic queries ϕ, there is no possibility of a sound and complete proof procedure for the circumscriptive query problem itself, since the set $\{\langle \mathcal{R}, \phi, \psi \rangle \mid Circ(\mathcal{R}(P);P) \models \phi \Leftarrow \psi\}$ is not recursively enumerable.

However, all is not lost. In the following two sections of the paper, we will investigate two different techniques that are capable of solving the problem: "Why did the Mermaids get the sack?" In fact, as we will see at the end of Section 4, the mermaid example happens to belong to an interesting class of circumscriptive query problems that are decidable in polynomial time!

3 Prototypical Proofs

The definition in (8)–(9) has a relatively simple form: It consists of a nonrecursive Horn clause, (8), and a linear recursive Horn clause, (9). Let us call this a *linear recursive definition*. In this section we will show how to solve the circumscriptive query problem for linear recursive definitions, using a special type of

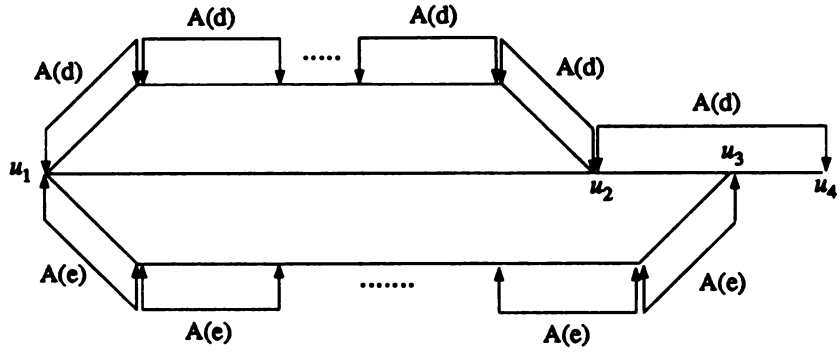


Figure 1: Why Did the Mermaids Get the Sack?

inductive proof. Our solution is a novel application of *second-order intuitionistic logic programming* [McCarty, 1988a; McCarty, 1988b; McCarty and Meyden, 1991; Nadathur and Miller, 1988; Miller, 1989].

The first step is to simplify the problem. Suppose we apply predicate completion to (8) alone, to obtain the following:

$$\mathbf{P}(x, t_1, t_2) \Rightarrow \mathbf{A}(x, t_1, t_2) \wedge t_1 < t_2 \tag{12}$$

We think of this as the *prototypical definition* of '$\mathbf{P}(x, t_1, t_2)$', and write it in general as '$P(\mathbf{x};\mathbf{t}) \Rightarrow P^0(\mathbf{x};\mathbf{t})$'. Let $\mathcal{P}(P)$ be the collection of prototypical definitions for the recursively defined predicates in $\mathcal{R}$. We now note the following simple fact, which is proven in [McCarty, 1992]:

Theorem 3.1 *If* $Circ(\mathcal{R}(P); P) \models \phi \Leftarrow \psi$ *then* $Comp(\mathcal{R}) \cup \mathcal{P}(P) \models \phi \Leftarrow \psi$.

Intuitively, if ϕ follows from all possible expansions of ψ, then it surely follows from the prototypical expansion of ψ. A solution to the right-hand side of Theorem 3.1 is called a *prototypical proof*. If we fail to find a prototypical proof, we have failed, period. But if we succeed, we can then use the prototypical proof to suggest an inductive proof for the left-hand side of Theorem 3.1.

Finding a prototypical proof is a simple problem in first-order intuitionistic logic programming [McCarty, 1988a; McCarty, 1988b; Nadathur and Miller, 1988]. The goal, $\phi \Leftarrow \psi$, is a universally quantified implication, so we can prove it by replacing all the universally quantified variables with unique constants, asserting the antecedent, ψ, into a hypothetical database, and then showing that the conclusion, ϕ, can be proven from this hypothetical database. In the mermaid example, we assert:

$$\mathbf{P}(\mathrm{d}, \mathrm{u}_1, \mathrm{u}_2), \mathbf{A}(\mathrm{d}, \mathrm{u}_2, \mathrm{u}_4), \mathbf{P}(\mathrm{e}, \mathrm{u}_1, \mathrm{u}_3), \quad \mathrm{u}_1 < \mathrm{u}_2 < \mathrm{u}_3 < \mathrm{u}_4 \tag{13}$$

and try to prove (10) using (8)–(9) and (12). It is easy to see that this query succeeds with $x = \mathrm{d}$, $y = \mathrm{e}$, and $v_i = \mathrm{u}_i$ for $i = 1, 3$. Notice, however, that we need to use (12) twice in this proof, once for each instance of '**P**' in (13). Without the use of (12), we would only be able to prove that

$$\mathbf{A}(\mathrm{d}, \mathrm{u}_1, \mathrm{u}_2), \mathbf{A}(\mathrm{d}, \mathrm{u}_2, \mathrm{u}_4), \mathbf{A}(\mathrm{e}, \mathrm{u}_1, \mathrm{u}_3), \quad \mathrm{u}_1 < \mathrm{u}_2 < \mathrm{u}_3 < \mathrm{u}_4 \tag{14}$$

implies (10). Our task, therefore, is to "strengthen" the proof that (14) implies (10) into a proof that (13) implies (10).

There are two ways to try to do this, and each approach can be tried on each of the two instances of '**A**' generated by (12) from (13). We thus encounter a nondeterministic choice in the computation, and we will simply describe here the successful branch. The first approach uses predicate completion, and tries to finish the proof by means of a *disjunctive splitting procedure* [Loveland, 1991]. Let us apply this technique to prove that

$$\mathbf{A}(\mathrm{d}, \mathrm{u}_1, \mathrm{u}_2), \mathbf{A}(\mathrm{d}, \mathrm{u}_2, \mathrm{u}_4), \mathbf{P}(\mathrm{e}, \mathrm{u}_1, \mathrm{u}_3), \quad \mathrm{u}_1 < \mathrm{u}_2 < \mathrm{u}_3 < \mathrm{u}_4 \tag{15}$$

implies (10). The completion of (8)–(9) is:

$$\begin{aligned} &\mathbf{P}(x, t_1, t_2) \Rightarrow \\ &\quad \mathbf{A}(x, t_1, t_2) \wedge t_1 < t_2 \\ &\quad \vee (\exists t)[\mathbf{A}(x, t_1, t) \wedge \mathbf{P}(x, t, t_2) \wedge t_1 < t < t_2] \end{aligned} \tag{16}$$

We have already proven the desired result from the first disjunct in (16), and we leave it to the reader to show that the desired result also follows from the second disjunct. The proof requires the use of the linear order constraint: $t < \mathrm{u}_2 \vee t = \mathrm{u}_2 \vee \mathrm{u}_2 < t$, which in turn requires another use of the disjunctive splitting procedure. (The reader may wish to consult Figure 1 to see why this is so.)

The second approach is inductive. Returning to the original definition of '$\mathbf{P}(x, t_1, t_2)$', let us apply predicate completion to (9) alone:

$$\mathbf{P}(x, t_1, t_2) \Rightarrow \quad (17)$$
$$(\exists t)[\mathbf{A}(x, t_1, t) \wedge \mathbf{P}(x, t, t_2) \wedge t_1 < t < t_2]$$

and then replace all the instances of '**P**' in (17) with a predicate variable 'X':

$$X(x, t_1, t_2) \Rightarrow \quad (18)$$
$$(\exists t)[\mathbf{A}(x, t_1, t) \wedge X(x, t, t_2) \wedge t_1 < t < t_2]$$

We call this second-order sentence the *transformation* associated with '$\mathbf{P}(x, t_1, t_2)$', and we write it in general as '$X(\mathbf{x};\mathbf{t}) \Rightarrow \Delta X(\mathbf{x};\mathbf{t})$'. Thus '$\Delta X(x, t_1, t_2)$' denotes the right-hand side of (18). Although (18) is written as an implication, it should be viewed, quite literally, as an operation that *transforms* any relation between **x** and **t** that matches its left-hand side into a relation between **x** and **t** that matches its right-hand side. In particular, it can be used to transform the prototypical relation '$P^0(\mathbf{x};\mathbf{t})$', and then used again to transform the resulting relation, and so on. Intuitively, if the proof of ϕ is "preserved" under all such transformations, then ϕ must be true.

To formalize this argument, we need an induction schema. Notice, in "strengthening" the proof of (10) from (14) into a proof of (10) from (15), that we have actually shown the following:

$$\phi \Leftarrow \mathbf{A}(\mathrm{d}, u_1, u_2) \wedge \mathbf{A}(\mathrm{d}, u_2, u_4) \wedge \mathbf{P}(\mathrm{e}, u_1, u_3) \wedge \quad (19)$$
$$u_1 < u_2 < u_3 < u_4.$$

Let Φ be the schema common to (11) and (19) in which the first occurrence of '**P**' in (11) is taken to be the variable. Thus (11) would be written $\Phi(\mathbf{P})$, and (19) would be written $\Phi(\mathbf{P}^0)$. Substituting 'X' and 'ΔX' into Φ as well, we have:

Definition 3.2 *The induction schema for $\Phi(P)$ is the following sentence in second-order intuitionistic logic:*

$$\Phi(P) \Leftarrow \Phi(P^0) \wedge (\forall X)[\Phi(\Delta X) \Leftarrow \Phi(X)]$$

We now note a surprising fact: The fragment of second-order intuitionistic logic that is needed for the application of this induction schema has a complete proof procedure (see [McCarty, 1992; Miller, 1989]). The procedure is similar to the first-order proof procedure for universally quantified implications discussed in [McCarty, 1988b]. To prove the second conjunct on the right-hand side of Definition 3.2, we replace the predicate variable 'X' with a new predicate constant '!X', we assert Φ(!X) into the rulebase, and we try to prove $\Phi(\Delta!\mathrm{X})$. If this proof succeeds, then we have proven the goal: $(\forall X)[\Phi(\Delta X) \Leftarrow \Phi(X)]$.

Let us see how this works in the mermaid problem. Carrying out the indicated substitutions, Φ(!X) is:

$$\phi \Leftarrow !\mathrm{X}(\mathrm{d}, u_1, u_2) \wedge \quad (20)$$
$$\mathbf{A}(\mathrm{d}, u_2, u_4) \wedge \mathbf{P}(\mathrm{e}, u_1, u_3) \wedge$$
$$u_1 < u_2 < u_3 < u_4,$$

and $\Phi(\Delta!\mathrm{X})$ is equivalent to:

$$\phi \Leftarrow \mathbf{A}(\mathrm{d}, u_1, t) \wedge !\mathrm{X}(\mathrm{d}, t, u_2) \wedge \quad (21)$$
$$\mathbf{A}(\mathrm{d}, u_2, u_4) \wedge \mathbf{P}(\mathrm{e}, u_1, u_3) \wedge$$
$$u_1 < t < u_2 < u_3 < u_4,$$

so we assert (20) and try to prove (21). But (21) is itself a first-order universally quantified implication, so we assert the right-hand side of (21), with variables replaced by constants, and try to prove the left-hand side. We leave it to the reader to verify that this proof goes through. Note, in particular, that rule (20) is essential to this proof, and that the predicate '!X' in (20) unifies with the predicate '!X' in (21). Putting all these pieces together – if we assume the induction schema in Definition 3.2 – we have just succeeded in proving $\Phi(\mathbf{P})$, which is simply an abbreviation of the goal $\phi \Leftarrow \psi$ in (11).

The justification for our procedure is the following soundness theorem for inductive proofs (see [McCarty, 1992]). Let $\mathcal{S}(P)$ be the set of all induction schemata, i.e., the schemata given by Definition 3.2 for all linear recursive definitions in $\mathcal{R}$ and all possible Φ . Then:

Theorem 3.3 *If $Comp(\mathcal{R}) \cup \mathcal{S}(P) \models \phi \Leftarrow \psi$ then $Circ(\mathcal{R}(P); P) \models \phi \Leftarrow \psi$.*

It is interesting to compare Theorem 3.1 with Theorem 3.3. If we could always "strengthen $\mathcal{P}(P)$ to $\mathcal{S}(P)$, then we would have a sound and complete proof procedure for the circumscriptive query problem. But we know from Theorem 2.3 and Theorem 2.4 that this is impossible. Thus the construction of an appropriate induction schema in Definition 3.2 is necessarily a heuristic step.

In the following section of the paper, we will establish a stronger result, for a smaller class of problems.

4 A Decision Procedure

We have seen in Theorem 2.3 that when dealing with order relations, the circumscriptive query problem is not decidable, even if we restrict the query ϕ to be basic, whereas this problem was decidable in the absence of order relations (Theorem 2.2). We will show in this section that it is possible to recover decidability of basic queries containing order relations, provided we impose a constraint on the rules $\mathcal{R}$. The program $\mathcal{R}$ is said to be *regular* if every rule in $\mathcal{R}$ is a linear rule of the form:

$$\mathbf{Q}(\mathbf{x}; t_1, t_k) \Leftarrow \bigwedge B_i(\mathbf{x}, \mathbf{y}; t_1, \ldots, t_{k-1}) \wedge \quad (22)$$
$$\mathbf{R}(\mathbf{x}, \mathbf{y}; t_{k-1}, t_k) \wedge$$
$$t_1 < \ldots < t_{k-1} < t_k$$

in which the B_i are basic actions and **R** is an abstract action, or a rule of the same form except that the abstract action **R** is omitted. Note that (8)–(9), the set

of action definitions in the mermaid problem, is regular. The main result of this section is that if the rules $\mathcal{R}$ are regular, the the circumscriptive query problem is decidable for basic queries.

The most important feature of regular action definitions is the fact that the basic actions in the body of each recursive rule occur *prior* to the abstract action. Our decision procedure exploits this fact by interleaving two operations: (i) an expansion of the abstract actions using predicate completion, and (ii) a topological sort of the order constants in each expansion. Intuitively, we are searching for a model of ψ in which ϕ does not hold, i.e., a countermodel to the implication $\phi \Leftarrow \psi$. If no such countermodel exists, then $\phi \Leftarrow \psi$ is entailed by $Circ(\mathcal{R}(P); P)$. Although the models of ψ are potentially infinite in number, it turns out that we can determine the entailment of a basic query by checking only a *finite* set of *partial* expansions. (This approach will not work for general queries, but note that we can always expand a query containing only *nonrecursive* abstract actions into an equivalent basic query.)

We will illustrate our decision procedure with the mermaid example. The procedure is nondeterministic, and we will only explore one of the branches of the search tree. As in the proof procedure of Section 3, our starting point is the database shown in (13), which happens to be topologically sorted. Let us add a marker '||' to this database, and stipulate that all the actions to the left of the marker must be basic and linearly ordered. Since 'u_1' is the unique minimal element in the linear order, we can immediately move it to the left:

$$u_1 \;\|\; \mathbf{P}(d, u_1, u_2), \mathbf{A}(d, u_2, u_4), \mathbf{P}(e, u_1, u_3), \quad u_2 < u_3 < u_4, \tag{23}$$

We now have two abstract actions on the right that contain u_1, which is the maximal element in the linear order on the left, so we cannot shift the marker further until we have expanded these abstract actions. Let us use (16) for the expansion, choosing a disjunct nondeterministically. Suppose we select the second disjunct in each case. Then the two abstract actions in (23) are replaced by:

$$\mathbf{A}(d, u_1, t), \mathbf{P}(d, t, u_2), u_1 < t < u_2$$

and

$$\mathbf{A}(e, u_1, t'), \mathbf{P}(e, t', u_3), u_1 < t' < u_3$$

Our database is now only partially ordered, and there are two minimal elements, t and t', on the right side of the marker. Suppose we continue the topological sort by setting $t = t'$ and mapping both of these elements to the next point on the left side of the marker. We then obtain:

$$\mathbf{A}(d, u_1, t), \mathbf{A}(e, u_1, t), u_1 < t \;\| \tag{24}$$
$$\mathbf{P}(d, t, u_2), \mathbf{A}(d, u_2, u_4), \mathbf{P}(e, t, u_3), \quad u_2 < u_3 < u_4$$

The two basic actions '$\mathbf{A}(d, u_1, t)$' and '$\mathbf{A}(e, u_1, t)$' have been shifted to the left here, since they are now completely contained within the sorted segment of the database.

Is it necessary to continue the search further along this branch? Notice that the constant 'u_1' in (24) occurs only on the left side of the marker. Therefore, however the process of expanding and sorting is continued, the two atoms on the left will be the only atoms containing the constant 'u_1'. Can these atoms contribute in any way to the satisfaction of the query? Clearly they do not satisfy the query on their own. Thus, if the query is ever to be satisfied using these atoms, we would need to introduce at least one more atom containing the constant 'u_1', and we have just argued that this cannot happen. Thus, the two atoms on the left can make no contribution to the satisfaction of the query, and can safely be deleted. This leaves us with the database:

$$t \;\|\; \mathbf{P}(d, t, u_2), \mathbf{A}(d, u_2, u_4), \mathbf{P}(e, t, u_3), \quad u_2 < u_3 < u_4 \tag{25}$$

Notice that (25) is isomorphic to (23), which means that our procedure has looped. We can thus safely terminate the present branch of our search, and backtrack to consider alternative expansions and alternative topological sorts. We leave it to the reader to verify that the remaining branches eventually entail the query.

Although our argument for terminating the search in this example may seem *ad hoc*, it can actually be extended into a systematic procedure. This is done by means of an equivalence relation on the left halves of the partial expansions, which we now describe informally. Notice that the left halves interact with the right halves only through the constants in common to the two halves. Every model produced by continuing the expansion process from a partial expansion $L \| R$ can be written as $L \cup S$, where the order constants of S are linearly ordered and "to the right" of L, except, possibly, for the constants C in both L and R. Consider two left halves L and L', which share only constants C with the right halves. Say L and L' are equivalent with respect to ϕ if for every corresponding right half S of a model, $L \cup S$ satisfies ϕ if and only if $L' \cup S$ does. We show in [Meyden, 1992 forthcoming] that for regular rules this yields an equivalence relation of finite rank on the left halves of the partial expansions, so that these can be compactly encoded using the representatives of these equivalence classes. This reduces the query problem to checking a finite number of partial expansions, and formalizes the type of loop checking illustrated above.

This analysis leads to the following result:

Theorem 4.1 *If $\mathcal{R}$ is a set of regular action definitions and if ϕ is a basic query, then the problem $Circ(\mathcal{R}(P); P) \models \phi \Leftarrow \psi$ is decidable in deterministic space $O\left((|\psi| \cdot |\mathcal{R}|)^{4|\phi|}\right)$.*

Notice that the space complexity of the query problem depends exponentially on the size of ϕ, but is polynomial as a function of the size of the hypothetical assertion ψ and the rules $\mathcal{R}$. Thus, if we take ϕ to be fixed and view ψ and $\mathcal{R}$ as "data", then the *data complexity* of our decision procedure is in PSPACE. In the mermaid problem, this would correspond to taking the test for dismissal, ϕ, as fixed, and analyzing the implications of various scenarios as Sam returns from his nap.

This complexity can be reduced to PTIME by a further reasonable restriction. Say that a conjunction of abstract actions ψ has *k-bounded concurrency* if every model of ψ obtained by expanding abstract actions using the rules and topologically sorting satisfies the following property:

> For every point t there exist no more than k basic actions $A(\mathbf{x}, u, v)$ with $u < t < v$.

Although this condition refers to an infinite set of expansions, it turns out to be easily decidable, as shown in [Meyden, 1992 forthcoming]. We then have the following result:

Theorem 4.2 *For a fixed basic query ϕ and for abstract actions ψ that have k-bounded concurrency and contain no more than a fixed number of object constants, $Circ(\mathcal{R}(P); P) \models \phi \Leftarrow \psi$ is decidable in polynomial time.*

Note that our mermaid problem has 2-bounded concurrency and contains only two object constants, and thus satisfies the conditions of Theorem 4.2 with $k = 2$.

Theorem 4.1 can be generalized in a variety of ways. As we have already mentioned, decidability still holds for queries ϕ containing non-recursive defined predicates. Decidability is also preserved if one allows in the body of regular rules defined predicates whose (possibly recursive) definition does not contain order variables. We will use the latter extension in the next section. We note, however, that the complexity result stated in Theorem 4.2 does not apply to these extensions.

5 Countermodels as Plans

The mermaid problem is not, strictly speaking, a planning problem. We can recast it as a planning problem by asking whether there is any way for Ethel and Daphne to bring about the scenario depicted in Figure 1 without violating Sam's instructions. The inductive proof procedure in Section 3 and the decision procedure in Section 4 then show that such a "plan" is impossible. However, this is an artificial interpretation of the story.

In this section, we will analyze a second example in which a countermodel to the implication $\phi \Leftarrow \psi$ is easily seen to be a plan for "doing ψ and avoiding ϕ." The example also shows how the "closed world assumption" can be incorprated into our formalism. Consider the following story:

The Radioactive Robot

Two robots, R1 and R2, inhabit a suite of six rooms, connected as indicated in Figure 2. Initially, robot R1 is in room D and robot R2 is in room F. We wish to swap the locations of the robots. However, robot R2 is radioactive, and if the two robots are ever in the same room at the same time, R1 will also become contaminated with radiation. It is therefore necessary to ensure that the two robots are never in the same room at the same time.

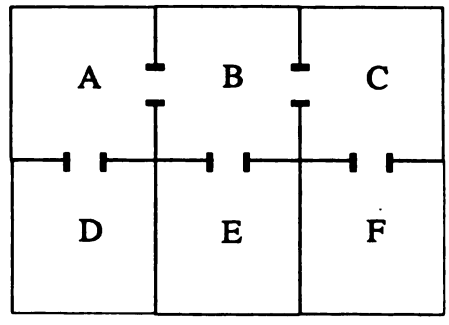


Figure 2: The Radioactive Robot

To formalize this problem, we introduce a basic predicate '$\text{In}(x, y, t_1, t_2)$' which asserts that x is in room y for a period of time starting at t_1 and ending at t_2. We also use an abstract action '$\textbf{Move}(x, y, z, t_1, t_2)$' which asserts that x moves from room y to room z over the time interval from t_1 to t_2. We let the constants A, ..., F denote the individual rooms. We then define ψ to be the following conjunction:

$$\textbf{Move}(\text{R1}, \text{D}, \text{F}, u_1, u_2) \wedge \textbf{Move}(\text{R2}, \text{F}, \text{D}, u_1, u_2) \tag{26}$$

and define ϕ to be the following existential formula:

$$(\exists y, t_1, t_2, t_1', t_2', t) [\text{In}(\text{R1}, y, t_1, t_2) \wedge \text{In}(\text{R2}, y, t_1', t_2') \wedge t_1 \leq t \leq t_2 \wedge t_1' \leq t \leq t_2']. \tag{27}$$

Clearly, ϕ describes the situation that we want to avoid, i.e., a situation in which R1 and R2 are both in the same room at the same time. Thus a countermodel

to the implication $\phi \Leftarrow \psi$, if it exists, would give us a way to swap the locations of the two robots without violating the stated constraint.

We now define the predicate '**Move**(x, y, z, t_1, t_2)'. Let $\mathcal{R}(P)$ consist of the rules:

$$\begin{aligned}&\textbf{Move}(x, y, z, t_1, t_2) \Leftarrow \\ &\quad \textbf{In}(x, y, t_1, t) \wedge \textbf{In}(x, z, t, t_2) \wedge \\ &\quad \textbf{Conn}(y, z) \wedge t_1 < t < t_2,\end{aligned} \tag{28}$$

$$\begin{aligned}&\textbf{Move}(x, y, z, t_1, t_2) \Leftarrow \\ &\quad \textbf{In}(x, y, t_1, t) \wedge \textbf{Move}(x, u, z, t, t_2) \wedge \\ &\quad \textbf{Conn}(y, u) \wedge t_1 < t < t_2,\end{aligned} \tag{29}$$

together with the following rules to express the connectedness relations of Figure 2:

$$\begin{aligned}&\textbf{Conn0}(\text{A}, \text{B}) \Leftarrow \\ &\textbf{Conn0}(\text{B}, \text{C}) \Leftarrow \\ &\textbf{Conn0}(\text{D}, \text{A}) \Leftarrow \\ &\textbf{Conn0}(\text{E}, \text{B}) \Leftarrow \\ &\textbf{Conn0}(\text{F}, \text{C}) \Leftarrow\end{aligned} \tag{30}$$

To ensure the symmetry of connectedness, we add the rules:

$$\begin{aligned}&\textbf{Conn}(x, y) \Leftarrow \textbf{Conn0}(x, y) \\ &\textbf{Conn}(x, y) \Leftarrow \textbf{Conn0}(y, x)\end{aligned} \tag{31}$$

We now let $P = \langle$**Move**, **Conn**, **Conn0**$\rangle$ be the tuple of predicates in the circumscription of $\mathcal{R}(P)$.

It is the circumscription of **Conn** and **Conn0** which captures the closed world assumption for the rooms in Figure 2. For example, predicate completion applied to (30) yields:

$$\begin{aligned}\textbf{Conn0}(x, y) \Rightarrow\ &[x = \text{A} \wedge y = \text{B}] \vee \\ &[x = \text{B} \wedge y = \text{C}] \vee \\ &[x = \text{D} \wedge y = \text{A}] \vee \\ &[x = \text{E} \wedge y = \text{B}] \vee \\ &[x = \text{F} \wedge y = \text{C}]\end{aligned} \tag{32}$$

However, to complete the specification of our problem, we need to add the "unique names" assumption for the constants A, ..., F. Formally, we do this by adding the theory *UN* which contains the fact '$c_i \neq c_j$' for each pair of distinct *object constants* c_i and c_j. Note that we do not apply the "unique names" assumption to the order constants, nor to the skolem constants generated by expanding abstract actions. Our query problem can now be formulated as follows:

$$UN \cup \mathit{Circ}(\mathcal{R}(P); P) \models \phi \Leftarrow \psi \tag{33}$$

The inductive proof procedure and the decision procedure of Sections 3 and 4 can both be adapted to handle this slight modification of our original query problem.

To understand this formalisation, consider the abstract action '**Move**(R2, F, D, u_1, u_2)'. Using rule (29), this action expands to:

$$\begin{aligned}&\textbf{In}(\text{R2}, \text{F}, u_1, t_1), \textbf{Move}(\text{R2}, x_1, \text{D}, t_1, u_2), \\ &\quad \textbf{Conn}(\text{F}, x_1), u_1 < t_1 < u_2,\end{aligned} \tag{34}$$

where x_1 and t_1 are skolem variables. Expanding the **Move** action in (34) using rule (28), we obtain:

$$\begin{aligned}&\textbf{In}(\text{R2}, \text{F}, u_1, t_1), \\ &\quad \textbf{In}(\text{R2}, x_1, u_1, t_2), \textbf{In}(\text{R2}, \text{D}, t_2, u_2), \\ &\quad \textbf{Conn}(\text{F}, x_1), \textbf{Conn}(x_1, \text{D}), u_1 < t_1 < t_2 < u_2.\end{aligned} \tag{35}$$

Now notice that if we were to expand the fact **Conn**(F, x_1) in (35) using **Conn0**(A,B), we would generate the equalities F = A and x_1 = B. But *UN* contains the assertion 'F $\neq$ A', and therefore this expansion implies a contradiction. In fact, the only expansion of **Conn**(F, x_1) that does not result in an immediate contradiction is one in which x_1 = C. However, this means that the second **Conn** fact in (35) now becomes **Conn**(C, D). Consistently expanding this fact turns out to be impossible, since every expansion of **Conn**(C, D) generates an equality that contradicts the "unique names" assumption.

In short, because the rules for **Conn0** contain only constants, every expansion of a fact **Conn**(x, y) will "ground out" the variables x and y in one of the constants A, ..., F. Thus the consistent expansions of an action **Move**(R, x_1, x_n, t_1, t_n) have the form

$$\begin{aligned}&\textbf{In}(\text{R}, c_1, t_1, t_2), \\ &\textbf{In}(\text{R}, c_2, t_2, t_3), \\ &\vdots \\ &\textbf{In}(\text{R}, c_{n-1}, t_{n-1}, t_n), \\ &t_1 < t_2 < t_3 < \ldots < t_n\end{aligned}$$

where $c_1, c_2, \ldots, c_n$ is a sequence of rooms A, ..., F such that for each i, c_i and c_{i+1} are connected according to Figure 2. For example, the goal ψ for our problem has an expansion in which robot R1 moves through rooms D,A,B,C,F, starting in room D at time 'u_1' and switching rooms at times $t_1 < \ldots < t_4$, and robot R2 moves through rooms F,C,B,A,D, starting in room F at time 'u_1' and switching rooms at times $t'_1 < \ldots < t'_4$. This expansion is not yet a model, since the constants t_i and t'_j are only partially ordered. When we attempt to topologically sort this partial order, of course, we find that all the sorted models satisfy the condition ϕ that we wish to avoid. Hence this particular expansion does not yield a plan for swapping the locations of the two robots while ensuring that they are never in the same room at the same time.

However, the reader can readily verify that the expansion in which R1 moves through rooms D,A,B,E,B,C,F and R2 moves through rooms F,C,B,A,D does have

topological sorts that avoid the undesirable condition ϕ. We need only guarantee that the interval over which R2 is in room B is contained in the interval in which R1 is in room E. We have thus found an expansion of ψ that does not satisfy 'ϕ', i.e., we have found a countermodel to the implication '$\phi \Leftarrow \psi$'. On the other hand, if we altered the connectivity of the rooms by deleting the rule **Conn0**(E,B) from (30), then the planning problem would no longer be solvable. In this case, the formula '$\phi \Leftarrow \psi$' would be entailed by $UN \cup Circ(\mathcal{R}(P); P)$.

Since the decision procedure in Section 4 proceeds by attempting to construct a countermodel to the circumscriptive entailment relation in (33), it may readily be modified to output such a countermodel in case this entailment relation fails. This modification yields a procedure that solves the planning problem whenever a solution exists, and reports unsolvability whenever no solution exists. That is, the procedure is not only correct, but it also guarantees termination.

6 Discussion

We have presented in this paper a novel technique for reasoning about indefinite actions: We assume that abstract actions are defined over a linear temporal order by a set of Horn clauses, $\mathcal{R}$, and we ask whether a hypothetical implication, $\phi \Leftarrow \psi$, is entailed by the *circumscription* of the defined predicates in $\mathcal{R}$. If $\mathcal{R}$ is nonrecursive, this approach gives us an action language similar to the language proposed by Allen and Koomen [1983] and a set of inferences similar to the abductive inferences analyzed by Eshghi [1988]. However, if $\mathcal{R}$ is recursive, we have a significant increase in expressive power: We can define an action '$R(t_1, t_n)$' which consists of "doing the action $A(t_1, t_2)$ some finite number of times." Although the general problem $Circ(\mathcal{R}(P); P) \models \phi \Leftarrow \psi$ is not in RE, we have developed two techniques for solving this problem in cases that seem to be of practical importance: (1) an inductive proof procedure for linear recursive definitions, which is sound but not complete; and (2) a decision procedure, which works for basic queries and regular rules. We have also identified a natural condition under which our decision procedure has PTIME data complexity.

An alternative approach to the representation of "doing α some finite number of times" is provided by the $\alpha*$ construct of *dynamic logic* [Pratt, 1976; Harel, 1979]. It is interesting to note that our "regular rules" can express the class of "regular events" which corresponds to the action modality of dynamic logic, whereas ordinary temporal logic cannot express this class of events at all [Wolper, 1983]. (It is also interesting to note, in the AI literature, that Rosenschein's formulation of planning problems in dynamic logic [Rosenschein, 1981] omitted the $\alpha*$ construct!) Despite its expressiveness, dynamic logic has been criticized within the AI community for its rigid "change-of-state" semantics [Shoham and Goyal, 1988]. More closely related to our approach is the family of *process logics* proposed by Pratt [Pratt, 1979; Harel *et al.*, 1982], in which modal operators can refer to the total trajectory in the execution of an action.

We have also argued in this paper that our formalism for reasoning about indefinite actions permits an elegant statement of a certain type of planning problem: A countermodel to the hypothetical implication $\phi \Leftarrow \psi$ can be interpreted as a *plan* for "doing ψ and avoiding ϕ." The close relationship between circumscription and abduction has been studied by Konolige [1992], and our procedure for constructing countermodels could also be viewed as a procedure for generating abductive inferences. To see this, note that an expansion of the predicates in ψ could be viewed as an "explanation" of ψ in terms of the definitions in $\mathcal{R}$. Under this interpretation, the goal ϕ plays the role of a negative "integrity constraint" [Eshghi and Kowalski, 1989; Kakas and Mancerella, 1990], i.e., a sentence that must be false in the abductive explanation. (If n integrity constraints $\neg I_1, \ldots, \neg I_n$ must be satisfied, constructing an abductive explanation of ψ corresponds to finding a countermodel for the relation $Circ(\mathcal{R}(P);P) \models \phi \Leftarrow \psi$, where $\phi = I_1 \vee \ldots \vee I_n$.) The main feature that is lacking in these studies of abductive inference, however, is a procedure for determining whether an abductive explanation that satisfies the integrity constraints, i.e., a countermodel, is impossible. This is precisely the point of our inductive proof procedure in Section 3 and our decision procedure in Section 4.

Moreover, most abductive theories of planning [Eshghi, 1988; Missiaen, 1991; Denecker *et al.*, 1992] have adopted a different logical framework from the one presented here, a framework that formalizes the effects of primitive actions on the state of the world. Abductive reasoning is then used within this framework to implement a nonlinear goal-directed regression-style planner. By contrast, our approach assumes that many planning situations come equipped with standard (but abstract) actions that are known to achieve certain goals. The planning problem then reduces to the problem of choosing appropriate expansions of these abstract actions and finding an appropriate way to schedule concurrent events. Of course, this is not the complete AI planning problem. It appears that even common simplified representations of planning, such as the STRIPS formalism, cannot be conveniently encoded using only definite rules. However, ours is not a new view of planning. It dates back to the early work of Tate [1977] and Sacerdoti [1977]. This view seems to have been obscured in most recent *logical* theories of planning.

We have focussed in this paper on reasoning about abstract actions, and have considered plans to be coun-

termodels, i.e., sets of basic actions in which the time points are linearly ordered. However, our formalism is amenable to *nonlinear planning*, in which one deals with basic actions and partially ordered time points. A non-linear plan represents a collection of linear plans, namely those obtained by topologically sorting the partial order. When dealing with non-linear plans, one needs a variety of *query modalities*. For example, the planner needs to be able to calculate whether there is some refinement of the partial order under which the plan is guaranteed to succeed. This is not the case if the query ϕ to be avoided holds in all the linear plans corresponding to the non-linear plan. Depending on the precise formulation of queries and actions, this may be a nontrivial problem [Chapman, 1985; Dean and Boddy, 1988]. For our formalism, even if ψ contains no abstract actions, determining the validity of the query $\phi \Leftarrow \psi$ with respect to linear orders requires consideration of all topological sorts of the partial order stated in ψ, potentially a very large set. We refer the reader to [Meyden, 1992] for a study of the complexity of this problem, which finds surprisingly high complexities, although certain interesting cases are in PTIME.

To formulate a greater variety of planning problems, we need, in addition, various modalities over actions. We suggest the *deontic modalities* as an appropriate set: **P** (permitted, in the "free choice" sense), **F** (forbidden), **O** (obligatory), and their respective negations. We have studied these modalities in previous work [McCarty, 1983; McCarty, 1986; Meyden, 1990; Meyden, 1991]. In future work, we will show how to combine a system of deontic logic with our present techniques for the representation of indefinite actions, and how to apply such a system in legal domains [McCarty, 1989; Schlobohm and McCarty, 1989].

References

[Allen and Koomen, 1983] J. F. Allen and J. A. Koomen. Planning using a temporal world model. In *Proceedings, International Joint Conference on Artificial Intelligence*, pages 741–747, 1983.

[Allen, 1984] J.F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[Chapman, 1985] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1985.

[Clark, 1978] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum, 1978.

[Dean and Boddy, 1988] T. Dean and M. Boddy. Reasoning about partially ordered events. *Artificial Intelligence*, 36:375–387, 1988.

[Denecker *et al.*, 1992] M. Denecker, L. Missiaen, and M. Bruynooghe. Temporal reasoning with abductive event calculus. In *Proceedings of the European Conference on Artificial Intelligence*, 1992. to appear.

[Eshghi and Kowalski, 1989] K. Eshghi and R. Kowalski. Abduction compared with negation as failure. In G. Levi and M. Martelli, editors, *Logic Programming: Proceedings of the Sixth International Conference*, pages 234–254, Boston, MA, 1989. MIT Press.

[Eshghi, 1988] K. Eshghi. Abductive planning with event calculus. In R.A. Kowalski and K. Bowen, editors, *Logic Programming: Proceedings of the Fifth International Conference and Symposium*, pages 562–579. MIT Press, 1988.

[Georgeff and Lansky, 1985] M.P. Georgeff and A.L. Lansky. A procedural logic. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 516–523, 1985.

[Harel *et al.*, 1982] D. Harel, D. Kozen, and R. Parikh. Process logic: Expressiveness, decidability and completeness. *Journal of Computer and System Sciences*, 25, 1982.

[Harel, 1979] D. Harel. *First-Order Dynamic Logic.* Springer-Verlag, 1979. LNCS No. 68.

[Kakas and Mancerella, 1990] A.C. Kakas and P. Mancerella. Generalized stable models: A semantics for abduction. In *Proceedings of the European Conference on Artificial Intelligence*, pages 385–391, 1990.

[Konolige, 1992] K. Konolige. Abduction versus closure in causal theories. *Artificial Intelligence*, 53:255–272, 1992.

[Kowalski and Sergot, 1986] R.A. Kowalski and M.J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.

[Lifschitz, 1985] V. Lifschitz. Computing circumscription. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 121–127, 1985.

[Loveland, 1991] D.W. Loveland. Near-Horn Prolog and beyond. *Journal of Automated Reasoning*, 7(1):1–26, 1991.

[Manna and Waldinger, 1987] Z. Manna and R. Waldinger. How to clear a block: A theory of plans. *Jounal of Automated Reasoning*, 3:343–377, 1987.

[McCarthy, 1980] J. McCarthy. Circumscription: A form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[McCarty and Meyden, 1991] L.T. McCarty and R. van der Meyden. Indefinite reasoning with definite rules. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 890–896, 1991.

[McCarty, 1983] L.T. McCarty. Permissions and obligations. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 287–294, 1983.

[McCarty, 1986] L.T. McCarty. Permissions and obligations: An informal introduction. In A.A. Martino and F. Socci Natali, editors, *Automated Analysis of Legal Texts: Logic, Informatics, Law*, pages 307–337. Elsevier North-Holland, 1986. Also available as Rutgers Technical Report LRP-TR-19.

[McCarty, 1988a] L.T. McCarty. Clausal intuitionistic logic. I. Fixed-point semantics. *Journal of Logic Programming*, 5(1):1–31, 1988.

[McCarty, 1988b] L.T. McCarty. Clausal intuitionistic logic. II. Tableau proof procedures. *Journal of Logic Programming*, 5(2):93–132, 1988.

[McCarty, 1989] L.T. McCarty. A language for legal discourse. I. Basic features. In *Proceedings of the Second International Conference on Artificial Intelligence and Law*, pages 180–189. ACM Press, June 1989.

[McCarty, 1992] L.T. McCarty. Computing with prototypes. Technical report, Computer Science Department, Rutgers University, 1992. Submitted for publication.

[McDermott, 1978] D. McDermott. Planning and acting. *Cognitive Science*, 2:71–109, 1978.

[McDermott, 1982] D. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.

[Meyden, 1990] R. van der Meyden. The dynamic logic of permission. In *Proceedings of the Symposium on Logic in Computer Science*, pages 72–78, Philadelphia, 1990.

[Meyden, 1991] R. van der Meyden. A clausal logic for deontic action specification. In *Proceedings of the International Logic Programming Symposium*, pages 221–238, San Diego, 1991.

[Meyden, 1992] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 331–345, 1992.

[Meyden, 1992 forthcoming] R. van der Meyden. *The Complexity of Querying Indefinite Information: Defined relations, Recursion and Linear Order.* PhD thesis, Rutgers University, 1992 (forthcoming).

[Meyden, 1992 in press] R. van der Meyden. Recursively indefinite databases. *Theoretical Computer Science*, 1992 (in press). An earlier version of this paper appears in *ICDT'90: Third International Conference on Database Theory*, Springer LNCS No. 470, S. Abiteboul and P.C. Kanellakis, eds., pp. 364-378 (1990).

[Miller, 1989] D.A. Miller. Lexical scoping as universal quantification. In G. Levi and M. Martelli, editors, *Logic Programming: Proceedings of the Sixth International Conference*, pages 268–283, 1989.

[Missiaen, 1991] L. Missiaen. *Localized Abductive Planning with the Event Calculus.* PhD thesis, Katholieke Universiteit Leuven, 1991.

[Nadathur and Miller, 1988] G. Nadathur and D.A. Miller. An overview of λPROLOG. In *Proceedings, Fifth International Conference and Symposium on Logic Programming*, pages 810–827, 1988.

[Peirce, 1931] C.S. Peirce. *Collected Papers of Charles Sanders Peirce*, volume 6:522-528. Harvard University Press, 1931.

[Pratt, 1976] V. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proceedings, 17th IEEE Symposium on Foundations of Computer Science*, pages 109–121, 1976.

[Pratt, 1979] V. Pratt. Process logic. In *Proceedings, 6th ACM Symposium on Principles of Programming Languages*, pages 93–100, 1979.

[Rosenschein, 1981] S. Rosenschein. Plan synthesis: A logical perspective. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 331–337, 1981.

[Sacerdoti, 1977] E.D. Sacerdoti. *A Structure for Plans and Behaviour.* Elsevier, New York, 1977.

[Schlobohm and McCarty, 1989] D.A. Schlobohm and L.T. McCarty. EPS-II: Estate planning with prototypes. In *Proceedings of the Second International Conference on Artificial Intelligence and Law*, pages 1–10. ACM Press, June 1989.

[Shanahan, 1989] M. Shanahan. Prediction is deduction but explanation is abduction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1055–1060, 1989.

[Shoham and Goyal, 1988] Y. Shoham and N. Goyal. Temporal reasoning in artificial intelligence. In H. Shrobe, editor, *Exploring Artificial Intelligence.* Morgan Kaufmann, 1988.

[Shoham, 1987] Y. Shoham. *Reasoning about Change.* MIT Press, Boston, 1987.

[Tate, 1977] A. Tate. Generating project networks. In *Proceedings, International Joint Conference on Artificial Intelligence*, pages 888–839, 1977.

[Wolper, 1983] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72–93, 1983.

Representations for Decision-Theoretic Planning: Utility Functions for Deadline Goals*

Peter Haddawy
Department of EE & CS
University of Wisconsin–Milwaukee
Milwaukee WI 53201
haddawy@cs.uwm.edu

Steve Hanks
Department of CS&E, FR–35
University of Washington
Seattle WA 98195
hanks@cs.washington.edu

Abstract

Symbolic (AI) planning techniques and decision theory have complementary strengths and weaknesses. Symbolic planning provides a computational theory of plan generation, but under unrealistic assumptions: perfect information and control over the world and a restrictive model of actions and goals. Decision theory provides a normative model of choice under uncertainty, but offers no guidance as to how the planning options are to be generated.

Our goal is to integrate the two approaches into a theory that preserves decision theory's model of choice while exploiting the computational techniques of symbolic planners. In order to do so we must confront the relationship between a symbolic planner's goals and a decision-theoretic agent's utility function. This paper focuses on a particular class of goals: those involving a temporal deadline. We provide a formal definition for deadline goals, show how to represent partial goal satisfaction, demonstrate how corresponding utility functions can be built relative to those goals, and suggest how the goal information can be used to limit a planning algorithm's inference.

1 Introduction

Reasoning about and planning for an uncertain world raises both representational and algorithmic problems: we need to represent change, uncertainty, and value or utility, and to use those concepts to represent various plans of action. And given such a representation for the world and for plans that might be executed in the world, we further need an efficient way to generate plausible plans, anticipate their results, improve their performance, and choose the best option from among them.

Decision theory addresses the representational problem, providing a rational basis for choice under uncertainty. The framework starts with the agent's preferences over an abstract set of possible outcomes, and guarantees the existence of probability and utility functions such that acting to maximize expected utility is rational in the sense that it respects those preferences. Note that the planner need not explicitly perform the decision-theoretic analysis. It suffices that the planner act according to the recommendations that such an analysis would make.

Decision theory does not, however, constitute a theory of reasoning about plans. The theory does not provide a vocabulary for describing planning problems, a method for generating options, or a computational model for choosing among plan alternatives. It merely dictates a rational choice *given* such a capability[1].

The capabilities provided by symbolic planning and decision theory are therefore complementary: the former provides methods for representing planning problems and generating alternative plans in response to externally supplied goals (but under restricted conditions that don't include uncertainty); the latter provides a method for choosing among alternatives and a language that allows reasoning with uncertainty, but provides (1) no guidance in structuring planning

*Thanks to Mike Wellman for many useful comments. Haddawy was supported in part by a grant from the graduate school of the University of Wisconsin-Milwaukee. Hanks was supported in part by NSF Grant IRI–9008670.

[1]Decision analysis—the study of applying decision-theoretic methodology—addresses these limitations, but in a subjective and non-algorithmic fashion.

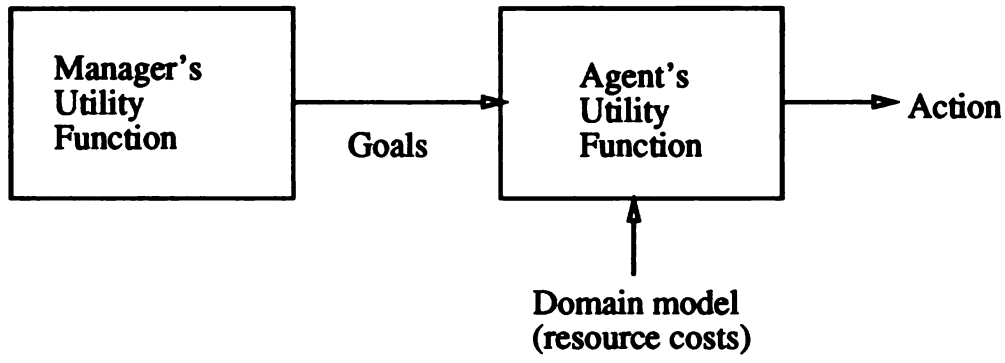


Figure 1: Goals as a means of communication

knowledge, (2) no way of generating alternatives, and more generally, (3) no computational model.

The first step toward integrating these two methodologies is to reconcile the two representations. Planning problems are couched in terms of

- a set of *operators* that effect change in the world,
- a description of some initial *world state* (usually expressed in some logical or quasi-logical language), and
- a *goal state description.*

Decision-theoretic problems are stated in terms of

- a probability distribution over possible outcomes and,
- utilities over those outcomes.

The relationship between the probabilistic model of the world and the operators and initial state has been studied as a problem of probabilistic temporal inference, *e.g.* [Hanks, 1990a] and [Haddawy, 1991a]. The relationship between the goal state description and an agent's utility model has received less attention.

We began analyzing this relationship in [Haddawy and Hanks, 1990], in which we articulated the similarities and differences between the two representations, gave an abstract characterization of a utility function expressing the notion of goal satisfaction, showed circumstances under which the problem of planning to achieve a goal state could be viewed as equivalent to planning to maximize a utility function, and presented several new forms for goal expressions that captured concepts like partial satisfaction and deadlines.

This paper extends the analysis by considering a particular class of goals: "deadline goals." We provide a formal account of these goals and show how utility functions can be easily specified relative to goals of that form. We then show how they can be integrated into a planning algorithm that can generate and refine plan options in a principled way.

We begin by summarizing our framework for expressing goals in a decision-theoretic framework.

2 The Role of Goals

Before we proceed with a formal analysis of goals and utilities we need to define more precisely what role these concepts will play in a planning system. [Haddawy and Hanks, 1990] advanced the idea that goals play two roles:

- they act as a device for communicating information about the planning problem and
- they act as a means of limiting inference in the planning process.

In the first case goals can be communicated more easily than utility functions, and in the second case the goals' symbolic content can aid the search for good plans.

Figure 1 makes the relationship more clear: let us assume some *manager*, who has a utility function over outcome states. He is designing an *agent*, and has a model of the agent's capabilities. The manager wants to communicate information to the agent that will cause it to act so as to increase the manager's utility; he uses goal expressions to communicate that information to the agent. The agent combines these goal expressions with other information it has about the world (*e.g.* some situation-independent information like the cost of various resources) to produce a utility function that guides its actions.

Our research explores what the form of these goal expressions should be, and how they can be assimilated into a utility function. We are particularly interested in the question of what constraints a goal expression should place on the agent's utility function.

3 Syntactic forms for goals

Our previous work [Haddawy and Hanks, 1990] began an analysis of symbolic goal expressions. We started

with the idea that goals should have a symbolic content as they do in classical AI planners, but added an explicit temporal component to the goal and also acknowledged the need for some representation of the utility associated with partial satisfaction of the goal.

3.1 The Language $\mathcal{L}_{tca}$

In order to talk about temporally qualified sentences in a probabilistic setting, we need a logic that can represent both time and probability. The logic of time, chance, and action $\mathcal{L}_{tca}$ is well suited to our purposes. A simplified version of the logic is described in [Haddawy, 1991b] and the full logic is described in [Haddawy, 1991a]. We describe here just that portion of the language relevant to this paper.

To refer to facts and event types occurring in time, the language defines two predicates:

- $HOLDS(FA, t_1, t_2)$ is true if fact FA holds over the time interval t_1 to t_2, and
- $OCCURS(EV, t_1, t_2)$ is true if event EV occurs during the interval t_1 to t_2.

Probability is treated as a sentential operator so it can be combined freely with other logical operators. We write $P_t(\phi)$ to denote the probability of a formula ϕ at time t. Although the language can represent the dynamics of probability over time by allowing the probability operator to be indexed by any time point, in this paper we will only index it by the current time (*now*). A model for $\mathcal{L}_{tca}$ consists of, among other things, a collection of chronicles and a probability measure over the chronicles. The probability of a sentence is defined as the measure of the set of chronicles in which the sentence is satisfied.

3.2 Goal expressions

We begin by breaking a goal into atemporal and temporal components. The former indicates *what* is to be achieved, the latter *when* it is to be achieved.

If ϕ is a formula containing only *HOLDS* and *OCCURS* predicates with temporal arguments t and t' then there are two forms for goals, the *existential* and the *universal*:

$$\exists t, t' \, (t_1 \leq t \leq t' \leq t_2) \wedge \phi$$
$$\forall t, t' \, (t_1 \leq t \leq t' \leq t_2) \rightarrow \phi$$

where ϕ is called the *atemporal component* and the rest of the formula is called the *temporal component*. This paper will be concerned only with the former type: existential or *deadline* goals. Examples are

- Have block A on block B by noon.

$$\exists t, t'(now \leq t \leq t' \leq noon) \wedge HOLDS(\text{on(A,B)}, t, t')$$

- Get all the rocks to the depot by noon.

$$\exists t, t'(now \leq t \leq t' \leq noon) \wedge \forall r(HOLDS(\text{rock(r)}, t, t') \rightarrow HOLDS(\text{loc(r,depot)}, t, t'))$$

Below we will provide some prototypical forms for expressing partial satisfaction for deadline goals, and also talk about the atemporal component in more detail. But first we will introduce our goal-oriented utility model.

4 The Form of Utility Functions

We express the agent's utility function at the top level as a weighted sum of the utilities associated with its top-level goals (each a formula as described above), plus a "residual utility" component:

$$\mathbf{U}(c) = \Sigma_{i=1}^{n} k_i \mathbf{UG}_i(c) + k_r \mathbf{UR}(c) \qquad (1)$$

The utility function is defined relative to a *chronicle* c, which represents one possible outcome of a course of events or plan[2].

A chronicle can also be assigned a probability, making the expected utility of a plan

$$\mathbf{EU}(\mathcal{P}) = \Sigma_c \mathbf{U}(c) \times \mathbf{P}(c|\mathcal{P}) \qquad (2)$$

where $\mathbf{P}(c|\mathcal{P})$ is the probability that chronicle c will be realized in the world, given that the agent is committed to executing plan $\mathcal{P}$. [Hanks, 1990b] discusses how to define and compute this probability.

Returning to Equation 1 we note two main components:

- the n utility functions $\mathbf{UG}_i$ associated with the agent's n top-level goals, and
- the residual utility function UR.

Equation 1's form implies that the agent's preferences over the component goals are *additive independent*: preferences over lotteries on the goals depend only on the marginal probability distributions for the goals and not on their joint probability distribution [Keeney and Raiffa, 1976, Sect 6.2]. In other words we assume that preference for a particular level of satisfaction for one goal does not depend on the extent to which the other goals are satisfied or on the level of residual utility.

This restriction means that we cannot represent conjunctive goals like "have the truck fueled and loaded by 7AM" as two separate top-level goals, since presumably satisfying either one without the other affords low utility but their conjunction affords high utility. We are free to represent them as a *single* goal, however,

[2] [McDermott, 1982] defines chronicles in terms of a temporal logic, and [Hanks, 1990b] and [Haddawy, 1991a] extend the notion to a probabilistic framework.

and this paper discusses the case where a top-level goal consists of a conjunction of symbolic subgoals that can interact.

The residual utility function takes into account any other resources produced or consumed in the course of achieving the goal. Assuming that residual utility is independent from goal utility implies that the cost of a unit of resource (say fuel) consumed in achieving a goal is the same regardless of *which* goal it was consumed in achieving and regardless of the extend to which all of the goals were satisfied.

We should note that the assumption of utility independence does not imply that that goals are *probabilistically* independent. Our assumption implies only that the *utility* derived from satisfying one top-level goal does not depend on the extent to which the other goals are achieved. As an extreme example, the agent may well have two conflicting goals P and $\neg P$, each with an associated utility function. Although the utilities are by assumption independent, the agent will discover that there is no chronicle in which both are satisfied, therefore the utilities for two goals cannot be realized simultaneously.

We will temporarily ignore the top-level utility function U(c) as well as the residual utility function UR(c), and concentrate on how to specify utility functions for individual goals (the $UG_i(c)$ functions).

5 Utility Functions for Individual Goals

We need to define the utility function for individual goals in a way that allows partial satisfaction for both the temporal and atemporal components. We motivate the development with an example.

Suppose we are employing a delivery agent whose task it is to deliver two tons of rocks to the depot by 2pm. He may require several trips to do so. How can we structure his pay in order to encourage him to meet our deadline as closely as possible? A reasonable way to reward the driver is to pay him in proportion to each quantity of rocks delivered on each trip, up to a total of two tons. The pay for each delivery would be discounted by an amount proportional to the amount of time by which each delivery misses the deadline. The amount the driver gets paid is then the sum of the rewards for all the deliveries he makes up to two tons. If the driver acts to maximize his expected pay, we will also be maximizing our utility for the goal.

Generalizing from this example, we can think of the pay structure as a utility function: we assign utility to a delivery of rocks and we weight the value of the delivery by a temporal coefficient representing how utility decays after the deadline. Each individual delivery is an event that changes some quantity associated with the atemporal goal component (in this case the quantity of rocks delivered so far). We therefore need to define utility in terms of changes in the value of the atemporal component. Following sections will show how this is done.

To derive the overall goal utility we sum the utilities for the individual delivery events, assuming that the utility associated with the increase in the number of rocks at the depot resulting from each delivery is independent of any other changes in the number of rocks that have occurred or will occur.

We will first show how to define utility over changes in the degree of satisfaction of the goal's atemporal component, then discuss the form of the temporal weighting coefficient, and finally present the utility function that results from combining the temporal and atemporal components.

5.1 Utility Functions for the Atemporal Component

The atemporal utility function represents the amount of utility associated with changes that affect the extent to which the goal's atemporal component is satisfied. We can view it as a function whose domain is the set of changes that can occur with respect to the atemporal component (*e.g.* changes in the number of rocks delivered to the depot or the number of symbolic goal conjuncts that are true). Notice that we cannot simply assign utility to changes since the atemporal utility must be bounded from above by a quantity representing complete satisfaction of the goal. Thus we will instead define it in terms of a degree of satisfaction function $\mathbf{dsa}_i$ that measures the degree to which the atemporal component is satisfied at a time in a chronicle. The degree of satisfaction is 0.0 if the atemporal component is completely unsatisfied and 1.0 if it is completely satisfied. If we assume that the degree of satisfaction of the atemporal component does not decrease with time over the planning horizon, we can define the atemporal utility associated with a change in the goal's state as the difference between the two **dsa** values.

This range of values will depend on the actual form of the goal expression, and in this paper we give two examples:

- symbolic attributes—a conjunction of symbolic propositions like "a big red cylinder on the table," and
- quantitative attributes—the value of a real- or integer-valued quantity like the truck's fuel level or the number of items in a box.

Although quantitative attributes could be defined in terms of a set of symbolic attributes, we can simplify the associated **dsa** function by exploiting the structure

of the quantitative attribute.

5.2 Symbolic Attributes

Here we consider how to define a function specifying the utility associated with partial satisfaction of symbolic-attribute goals. A symbolic attribute is any logical formula containing only HOLDS and OCCURS predicates. For example, the goal of having block A on top of a red block by noon would be represented as

$$\exists t, t' \, (now \leq t \leq t' \leq noon) \wedge \\ \exists x \; HOLDS(On(A, x), t, t') \wedge \\ HOLDS(Red(x), t, t')$$

We define the degree of satisfaction (**dsa**) function for a symbolic-attribute goal in terms of a sequence S of mutually exclusive and exhaustive formulas $(\sigma_1, \sigma_2, ..., \sigma_n)$ such that σ_n is the actual atemporal component of the goal and σ_i represents a greater degree of satisfaction than σ_j if $i > j$. We associate a degree of satisfaction value with each σ_i such that $\mathbf{dsa}(\sigma_1) = 0$ and $\mathbf{dsa}(\sigma_n) = 1$. (For brevity we will eliminate the *HOLDS* and *OCCURS* predicates in the specification of S.) So for the above example, S might be

i	σ_i	$\mathbf{dsa}(\sigma_i)$
1	$\neg\exists x On(A, x)$	0.0
2	$\exists x On(A, x) \wedge \neg Red(x)$	0.7
3	$\exists x On(A, x) \wedge Red(x)$	1.0

The simplest such function would be one that admits no partial satisfaction of the goal:

i	σ_i	$\mathbf{dsa}(\sigma_i)$
1	$\neg\exists x On(A, x) \wedge Red(x)$	0.0
2	$\exists x On(A, x) \wedge Red(x)$	1.0

The degree of satisfaction must actually be evaluated at a point in time within a chronicle. To do so we take the value to be the **dsa** value associated with the (unique) formula that is satisfied in that chronicle at that time. Since the formulas σ_i are mutually exclusive and exhaustive, we are guaranteed that exactly one of them will be true at each time within the chronicle.

We use this definition of **dsa** to define the utility relative to the atemporal component of a time in a chronicle as the difference between the degree of satisfaction at that time and the highest degree of satisfaction at any earlier time. The reason we use the highest degree of satisfaction of any earlier time is that we do not want to reward the agent for undoing then reaccomplishing the goal.

Defining the atemporal utility function in this way requires that the utility function is bounded and that all preferences for lotteries over changes in degree of satisfaction at any time t are the same as preferences for lotteries at any other time t'. The atemporal utility function so defined ranges between zero and one.

Notice that the table of σ_i's need not be complete in that it need not specify degree of satisfaction values for all possible changes. Missing entries indicate that some potential changes in degree of satisfaction are assigned zero utility. The second **dsa** function above provides an example: there is no utility associated with satisfying either conjunct in isolation; utility is recorded only when all conjuncts become true.

For some specialized types of symbolic goal structures the degree of satisfaction function may be defined more succinctly. For example, if the atemporal component is a conjunction of atomic formulas not sharing any variables, and if utility is additive in the changes in degree of satisfaction associated with each of the conjuncts, then we can define the degree of satisfaction of each of the conjuncts individually, and the utility associated with a time point in a chronicle is the sum of the utilities associated with each conjunct.

5.3 Quantitative Attributes

A quantitative attribute is a special kind of symbolic attribute: a logical formula containing only *HOLDS* predicates in which the first argument expresses equality or inequality between a term and a numeric quantity. An example of a quantitative-attribute deadline goal is to have two tons of rocks at the depot by noon:

$$\exists t, t' \, (now \leq t \leq t' \leq noon) \wedge \\ HOLDS((= \text{tons-rocks-at-depot } 2), t, t')$$

The degree of satisfaction function over such an attribute could in principle be defined in the same way as we did for symbolic attributes above, but such a definition would be unmanageably large. Since degree of satisfaction for quantitative attributes is simply a function of the quantity's magnitude, we can define it directly in terms of that magnitude. The degree of satisfaction for the rock-delivery example would simply be a function of the quantity of quantity of rocks at the depot (again evaluated relative to a time point in a chronicle).

With conjunctive quantitative attributes, things get a little more complicated. Suppose we have the goal of having 12,000 pounds of polystyrene, 480 pounds of colorant, and 120 pounds of UV stabilizer at a plastics manufacturing plant by time t_2:

$$\exists t, t' \, (now \leq t \leq t' \leq t_2) \wedge \\ HOLDS((= \text{poly } 12{,}000), t, t') \wedge \\ HOLDS((= \text{colorant } 480), t, t') \wedge \\ HOLDS((= \text{UV } 120), t, t')$$

These quantities were not chosen arbitrarily; they represent the quantities of materials necessary to manufacture 2000 units of a particular product. They need

to be present at the plant in a particular proportion in order to be useful. That is to say, only that quantity that is present in the right proportion can be used. This is a common characteristic of conjunctive quantitative attributes. Consequently, degree of satisfaction will be a function of the maximum amount of the materials that are present in the required proportion. In this case, the necessary proportion is 100:4:1. So to derive the degree of satisfaction of a time in a chronicle, we let

$$q = min(x/100, y/4, z)$$

where x,y, and z are the quantities of polystyrene, colorant, and UV stabilizer, respectively. Then if we require 6 pounds of polystyrene to manufacture one unit, the degree of satisfaction would be some nondecreasing function of $\lfloor 100q/6 \rfloor$, normalized to range between 0 and 1.

To summarize, we have motivated the need for a function that measures partial satisfaction of a goal's atemporal component, a function $\mathsf{DSA}_i(t, c)$ that measures the extent to which the goal is satisfied at a particular time point within a particular chronicle. This function is defined in a manner specific to the goal's type: for symbolic goals the programmer supplies an appropriate dsa function, for quantitative attributes he supplies a function that maps the quantity's value to a degree-of-satisfaction level.

5.4 The Coefficient Representing the Temporal Component

We have to represent partial satisfaction of the temporal component (deadline) as well, and we do so by weighting the DSA_i value by a temporal coefficient representing the extent to which the deadline was violated. The coefficient for the temporal component is a function CT_i of the deadline and a time point. A deadline goal states that we wish to achieve some state or attain some level of a quantity by a given time. We further assume that there is no value associated with achieving the goal prior to the deadline. If that were not the case we we could have specified an earlier deadline. So the temporal coefficient will 0.0 at all times prior to the deadline t_d. After the deadline its value will drop off as a function of time, representing the fact that we prefer to have the goal satisfied by the deadline. If we have a hard deadline, the value may be 1.0 at t_d but drop to 0.0 immediately. More generally, the value will be a monotonically nonincreasing function of time after the deadline—it need not be strictly decreasing. A flat region on the coefficient function curve over some time interval after the deadline represents indifference about which time in that region the goal is satisfied.

The reader may wonder why the temporal coefficient is not symmetric in that it does not increase up to the deadline then decrease as the deadline passes. To return to the rock-delivery example, we define the deadline as the first time at which the rocks actually become useful, and therefore before the deadline they have no intrinsic value: if the deadline is noon, then the only reason to deliver the rocks at any time prior to noon is so they will in fact be at their destination at the time they are to be used.

So why should early delivery be assigned lower utility? Two reasons come to mind:

- Because if the rocks are delivered early there is a chance that they will be stolen or otherwise be made unavailable before the deadline.
- Because if the rocks are delivered early they will have to be stored, which incurs some cost.

Our utility model covers both of these cases, but neither of them bears directly on the utility associated with the goal itself. In the first place, if there is some probability that early deliveries will disappear, then the plan projector (the module that generates possible outcomes and assigns them probabilities) should generate a plan outcome (chronicle) in which they disappear. In that chronicle the goal is not met at all, and therefore the plan will be assigned lower utility. The storage charge in the second case can be noted by the projector as well, but this cost is appropriately recorded in the residual utility function; it is not by rights a cost associated with the delivery goal itself.

6 Combining the Temporal Coefficient and the Atemporal Utility Function

So far we have defined the atemporal satisfaction function DSA_i and the temporal coefficient CT_i, both of which are functions over time points and chronicles. To form the utility function for a deadline goal (which is a function of the chronicle alone) we need to evaluate and combine DSA and CT values at selected time points within the chronicle.

We first make the following assumptions about changes in atemporal utility:

- A change in degree of satisfaction of the atemporal component at time t is utility independent of changes at any earlier or later time.
- There are a countable number of changes in the degree of satisfaction over the course of a chronicle.

Under these assumptions the appropriate expression for computing a goal's utility is an additive utility function:

$$\mathsf{UG}_i(c) = \mathsf{DSA}_i(c, t_D) + \sum_{\{t:\,(t>t_D)\}} (\mathsf{DSA}_i(t, c) - max_{t_D \le \hat{t} < t} \mathsf{DSA}_i(\hat{t}, c)) \cdot \mathsf{CT}_i(t) \quad (3)$$

where t_D is the time of the deadline. The basic form of this utility function is similar to that presented by Meyer [Keeney and Raiffa, 1976, Sect. 9.3.2] for the utility of a consumption stream. The only difference is that Meyer's formulation defines utility directly in terms of consumption—which is the change in the agent's wealth—while our dsa functions specify this change in utility indirectly. Another minor difference is that Meyer's formulation allows consumption to be negative, indicating a loss of wealth, whereas we are only interested in positive changes in degree of satisfaction[3].

Consider, for example, the goal to have two tons of rocks at the depot by noon. We can use one of two trucks. The big truck carries two tons in one load but is slow: it will get all two tons to the depot by 2pm. The small truck carries only one ton but is fast: it will get one ton there by 1pm and two tons there by 2pm. Which plan affords higher utility? The answer depends, of course, on the utility decay associated with missing the deadline compared to the utility decay associated with missing some rocks. Suppose that the degree of satisfaction of zero tons is 0.0, the degree of satisfaction of one ton is 0.5 and the degree of satisfaction of two tons is 1.0. Suppose further that the temporal coefficient is a linearly decreasing function that goes from 1.0 at noon to 0.0 at 3pm. So $\mathrm{ct}(t) = 1 - t/3$, where t is the number of hours past noon. Based on these values, the utility of the chronicle that results from using the big truck is $(1)(1/3) = 1/3$ and the utility of the chronicle that results from using the small truck is $(1/2)(2/3)+(1/2)(1/3) = 1/2$. Hence we prefer to use the small truck over the big truck. This preference fits our intuition since based on the specifications of the atemporal utility function and the temporal coefficient, we associate some benefit with having some portion of the two tons of rocks at the depot earlier.

7 Using the Utility Functions to Rank Plans

One of the main goals of the present work is to use information in the utility function's symbolic structure to guide the search for good plans, which generally will involve demonstrating that one plan is preferable to another. At worst establishing this relationship involves computing the expected utility of both plans, a process that requires generating all possible outcomes for each.

By exploiting the structure of the utility functions we may be able to establish the same relationships without performing the full expected-utility calculation. We do so by establishing relationships among the individual goals' symbolic components that indicate corresponding relationships among the utility functions. These relationships will typically take the form:

> Suppose that one of the agent's goals is g and that there are two formulas ϕ and ψ that bear some relation to g. More particularly, the truth of ϕ indicates a *high* goal-related utility whereas the truth of ψ indicates a *low* utility. Further suppose that there are two alternative plans, $\mathcal{P}_1$ and $\mathcal{P}_2$. If the probability that ϕ is true by some time t_1 given $\mathcal{P}_1$ is at least α, and if the probability that ψ is true for all times before t_2 given $\mathcal{P}_2$ is at least β, and if α exceeds some function of β, ϕ, ψ, t_1, and t_2, then $\mathcal{P}_1$'s expected utility is guaranteed to be greater than $\mathcal{P}_2$'s.

Having established a relationship of that form the planner need only establish two probability bounds associated with specific symbolic propositions in order to eliminate one plan from further consideration.

This technique will ultimately have two advantages:

1. it reduces the general problem of expected-utility calculations to the more specific task of deciding whether a particular probability exceeds a particular threshold, and
2. it allows us to perform the expected-utility analysis incrementally—at each stage we can eliminate some plans from consideration, again limiting the amount of inference necessary to choose a good course of action.

7.1 Bounds for Symbolic Attributes

We first look at establishing probability bounds for deadline goals with symbolic atemporal components. Let ϕ and ψ be two symbolic attributes as described above. Suppose our symbolic goal is

$$\exists t, t'(now \leq t \leq t' \leq t_1) \wedge \\ HOLDS(\mathrm{On(A,B)}, t, t') \wedge HOLDS(\mathrm{On(B,C)}, t, t')$$

and we define the dsa function as follows:

i	σ_i	$\mathrm{dsa}(\sigma_i)$
1	$\neg\mathrm{On(A,B)} \wedge \neg\mathrm{On(B,C)}$	0.0
2	$\mathrm{On(B,C)} \wedge \neg\mathrm{On(A,B)}$	0.5
3	$\mathrm{On(A,B)} \wedge \mathrm{On(B,C)}$	1.0

the two formulas might be

$$\phi = HOLDS(On(A,B), t, t')$$
$$\psi = \neg HOLDS(On(B,C), t, t').$$

Now suppose that for plan $\mathcal{P}_1$ we can establish that

$$P_{now}(\exists t, t'\,(t \leq t' < t_1) \wedge \phi \mid \mathcal{P}_1) \geq \alpha$$

[3] The expression for the continuous case is $\mathrm{DSA}_i(c, t_D) + \int_{t_D}^{\infty} \mathrm{CT}_i(t) \cdot \frac{d\mathrm{DSA}_i(c,t)}{dt} dt$.

and for plan $\mathcal{P}_2$

$$P_{now}(\forall t, t'\,(t \leq t' < t_2) \rightarrow \psi \mid \mathcal{P}_2) \geq \beta$$

Under what conditions can we say that plan $\mathcal{P}_1$ is preferable to plan $\mathcal{P}_2$? We must determine the lowest value of $\mathsf{EU}(\mathcal{P}_1)$ and the highest value of $\mathsf{EU}(\mathcal{P}_2)$ consistent with these two constraints. Let σ_ϕ be the formula of lowest dsa consistent with ϕ (in the example $\sigma_\phi = \sigma_3$). We are guaranteed the existence of such a formula since the σ_i are exhaustive. The expected utility of plan $\mathcal{P}_1$ is lowest if

1. with probability α *i)* σ_ϕ is achieved at time t_1 and *ii)* the goal is not partially achieved at any time earlier than t_1, and
2. with probability $1-\alpha$ the goal is completely unsatisfied.

So by Equation 4 we have

$$\mathsf{EU}(\mathcal{P}_1) \geq \alpha \cdot \mathsf{dsa}(\sigma_\phi) \cdot \mathsf{ct}(t_1) + (1-\alpha) \cdot 0$$

Now let σ_ψ be the formula of *highest* dsa consistent with conjunct ψ ($\sigma_\psi = \sigma_2$). The expected utility of plan $\mathcal{P}_2$ is highest if

1. with probability β *i)* σ_ψ is achieved by the deadline and *ii)* the goal is completely achieved immediately after time t_2, and
2. with probability $1-\beta$ the goal is completely satisfied by the deadline.

Again by Equation 4:

$$\mathsf{EU}(\mathcal{P}_2) \leq \beta[\mathsf{dsa}(\sigma_\psi) + (1 - \mathsf{dsa}(\sigma_\psi))\mathsf{ct}(t_2)] + (1-\beta)$$

And we therefore know that $\mathcal{P}_1$'s expected utility is higher than $\mathcal{P}_2$'s if

$$\alpha > \frac{\beta[\mathsf{dsa}(\sigma_\psi) + (1 - \mathsf{dsa}(\sigma_\psi))\mathsf{ct}(t_2)] + (1-\beta)}{\mathsf{dsa}(\sigma_\phi) \cdot \mathsf{ct}(t_1)}$$

The values for σ_ϕ and σ_ψ and the times t_1 and t_2 will determine how useful our probability bounds are. A low t_1 and a ϕ that is inconsistent with low utility σ_i's will give us a high lower bound on $\mathsf{EU}(\mathcal{P}_1)$. Similarly, a high t_2 and a ψ that is inconsistent with high utility σ_i's will give us a low upper bound on $\mathsf{EU}(\mathcal{P}_2)$.

7.2 Bounds for Quantitative Attributes

Now suppose that our goal is stated in terms of some quantity Q. Assume that the dsa is a monotonically increasing function of the quantitative attribute[4]. Suppose we know that for plan $\mathcal{P}_1$

$$P_{now}(\exists t, t'\,(t \leq t' < t_1) \wedge HOLDS((\geq Q\, k_1), t, t') \mid \mathcal{P}_1) \geq \alpha$$

and for plan $\mathcal{P}_2$

$$P_{now}(\forall t, t'\,(t \leq t' < t_2) \rightarrow HOLDS((\leq Q\, k_2), t, t') \mid \mathcal{P}_2) \geq \beta$$

(where k_1 and k_2 are constants). Under what conditions can we say that we prefer plan $\mathcal{P}_1$ to plan $\mathcal{P}_2$? The expected utility of $\mathcal{P}_1$ is lowest if

1. with probability α *i)* we achieve a level k_1 for Q at time t_1 and *ii)* Q has value zero at all times prior to t_1, and
2. with probability $(1-\alpha)$ Q has value zero at all times.

Under those circumstances we know that

$$\mathsf{EU}(\mathcal{P}_1) \geq \alpha \cdot \mathsf{dsa}(k_1) \cdot \mathsf{ct}(t_1)$$

The expected utility of plan $\mathcal{P}_2$ is highest if

1. with probability β *i)* Q attains level k_2 by the deadline and *ii)* the maximum possible value of Q is attained immediately after t_2, and
2. with probability $1-\beta$ the maximum possible value of Q is attained by the deadline.

Then we know that

$$\mathsf{EU}(\mathcal{P}_2) \leq \beta \cdot [\mathsf{dsa}(k_2) + (1 - \mathsf{dsa}(k_2)) \cdot \mathsf{ct}(t_2)] + (1-\beta),$$

and then $\mathcal{P}_1$ is preferred to $\mathcal{P}_2$ if

$$\alpha > \frac{\beta \cdot [\mathsf{dsa}(k_2) + (1 - \mathsf{dsa}(k_2)) \cdot \mathsf{ct}(t_2)] + (1-\beta)}{\mathsf{dsa}(k_1) \cdot \mathsf{ct}(t_1)}$$

7.3 Bounds for Strictly Ordered Atemporal Attributes

Consider a goal with a symbolic atemporal component attribute consisting of a conjunction of formulas that do not share any variables. Suppose that the conjuncts can be ordered such that each conjunct dominates those later in the sequence in the sense that achieving that conjunct is more important than achieving all those later in the sequence. For example, if our conjuncts are g_1, g_2, and g_3 our strictly ordered atemporal utility function would satisfy the constraint that satisfying g_1 is preferable to not satisfying it, regardless of whether g_2 or g_3 are satisfied:

$$\begin{aligned}
\mathsf{dsa}(g_1 \wedge g_2 \wedge g_3) &> \mathsf{dsa}(\neg g_1 \wedge g_2 \wedge g_3)\\
\mathsf{dsa}(g_1 \wedge \neg g_2 \wedge g_3) &> \mathsf{dsa}(\neg g_1 \wedge g_2 \wedge g_3)\\
\mathsf{dsa}(g_1 \wedge g_2 \wedge \neg g_3) &> \mathsf{dsa}(\neg g_1 \wedge g_2 \wedge g_3)\\
\mathsf{dsa}(g_1 \wedge \neg g_2 \wedge \neg g_3) &> \mathsf{dsa}(\neg g_1 \wedge g_2 \wedge g_3)\\
\\
\mathsf{dsa}(g_1 \wedge g_2 \wedge g_3) &> \mathsf{dsa}(\neg g_1 \wedge \neg g_2 \wedge g_3)\\
\mathsf{dsa}(g_1 \wedge \neg g_2 \wedge g_3) &> \mathsf{dsa}(\neg g_1 \wedge \neg g_2 \wedge g_3)\\
&\vdots\\
\mathsf{dsa}(g_1 \wedge \neg g_2 \wedge \neg g_3) &> \mathsf{dsa}(\neg g_1 \wedge \neg g_2 \wedge \neg g_3)
\end{aligned}$$

[4] A symmetric analysis can be performed for monotonically decreasing atemporal utility.

i	σ_i	dsa(σ_i)
1	¬Loc(me,home)	0.0
2	Loc(me,home) ∧ ¬Possess(me,thai-food) ∧ ¬Possess(me,beer)	0.4
3	Loc(me,home) ∧ ¬Possess(me,thai-food) ∧ Possess(me,beer)	0.5
4	Loc(me,home) ∧ Possess(me,thai-food) ∧ ¬Possess(me,beer)	0.8
5	Loc(me,home) ∧ Possess(me,thai-food) ∧ Possess(me,beer)	1.0

Figure 2: Example atemporal utility function

and likewise, given that g_1 is true, it's always preferable to satisfy g_2 regardless of g_3's state:

$$\begin{aligned}
\textsf{dsa}(g_1 \wedge g_2 \wedge g_3) &> \textsf{dsa}(g_1 \wedge \neg g_2 \wedge g_3) \\
\textsf{dsa}(g_1 \wedge g_2 \wedge \neg g_3) &> \textsf{dsa}(g_1 \wedge \neg g_2 \wedge g_3) \\
\textsf{dsa}(g_1 \wedge g_2 \wedge g_3) &> \textsf{dsa}(g_1 \wedge \neg g_2 \wedge \neg g_3) \\
\textsf{dsa}(g_1 \wedge g_2 \wedge \neg g_3) &> \textsf{dsa}(g_1 \wedge \neg g_2 \wedge \neg g_3)
\end{aligned}$$

Finally we note that if g_1 and g_2 are both true then g_3 is preferable to its negation:

$$\textsf{dsa}(g_1 \wedge g_2 \wedge g_3) > \textsf{dsa}(g_1 \wedge g_2 \wedge \neg g_3)$$

Notice that we did not specify that g_2 dominates g_3 in cases where g_1 is false. The reason is that we will often assign zero utility to all states in which the most important goal is not satisfied. For example, given the goal to have block A on top of a red cylinder by noon, it might do us no good to have obtained a red cylinder if block A is not on top of it.

For a utility function of this form, we can prune away suboptimal plans by considering each conjunct individually in the priority order dictated by the dsa values. Suppose we have two plans $\mathcal{P}_1$ and $\mathcal{P}_2$ such that for plan $\mathcal{P}_1$

$$P_{now}(\exists t, t' \, (t \leq t' < t_1) \wedge g_1 \mid \mathcal{P}_1) \geq \alpha$$

and for plan $\mathcal{P}_2$

$$P_{now}(\forall t, t' \, (t \leq t' < t_2) \rightarrow \neg g_1 \mid \mathcal{P}_2) \geq \beta.$$

Under what conditions can we say that plan $\mathcal{P}_1$ is preferable to plan $\mathcal{P}_2$? We have not specified what the probability of the other conjuncts of the atemporal component is, so a lower bound on the expected utility of $\mathcal{P}_1$ is

$$\textsf{EU}(\mathcal{P}_1) \geq \alpha \cdot \textsf{dsa}(g_1 \wedge_i \neg g_i) \cdot \textsf{ct}(t_1)$$

and an upper bound on the expected utility of plan $\mathcal{P}_2$ is

$$\begin{aligned}
&\textsf{EU}(\mathcal{P}_2) \leq \\
&\quad \beta \cdot [\textsf{dsa}(\neg g_1 \wedge_i g_i) + (1 - \textsf{dsa}(\neg g_1 \wedge_i g_i))\textsf{ct}(t_2)] + \\
&\quad (1 - \beta).
\end{aligned}$$

So plan $\mathcal{P}_1$ is preferred to plan $\mathcal{P}_2$ if

$$\alpha > \frac{\beta[\textsf{dsa}(\neg g_1 \wedge_i g_i) + (1 - \textsf{dsa}(\neg g_1 \wedge_i g_i))\textsf{ct}(t_2)] + (1 - \beta)}{\textsf{dsa}(g_1 \wedge_i \neg g_i)\textsf{ct}(t_1)}$$

These bounds are meaningful because the prioritization of the conjuncts of the atemporal component means that satisfying the first conjunct dominates satisfying all the others:

$$\textsf{dsa}(g_1 \wedge_i \neg g_i) > \textsf{dsa}(\neg g_1 \wedge_i g_i)$$

so we can eliminate some suboptimal plans by just using the probability of satisfying or not satisfying the first conjunct. Once this has been done, we can use the probability of satisfying the first and second conjuncts versus the probability of satisfying the first conjunct but not the second, and so forth. This means that we initially consider only the consequences of plans relative to the first conjunct, then relative to the first two conjuncts, and so on until we have incorporated all the conjuncts. At that point we can perform an expected utility analysis of the remaining plans.

This ability to consider the conjuncts sequentially has important consequences for the projection process. [Hanks, 1990b] presents a probabilistic projection algorithm demonstrating that:

- it can be much cheaper to project a plan with respect to a single proposition (like one of our goal conjuncts) than it is to reason about all of the plan's effects,
- it can be much cheaper to determine whether the probability of a proposition exceeds a specific threshold than it is to compute the exact value of that probability.

7.3.1 Example

Let's consider an example. Suppose our goal is to be at home by 6:00 with some Thai food and some beer:

$$\begin{aligned}
&\exists t, t' \, (\textit{now} \leq t \leq t' \leq 6{:}00) \wedge \\
&\quad \textit{HOLDS}(\text{Loc(me,home)}, t, t') \wedge \\
&\quad \textit{HOLDS}(\text{Possess(me,thai-food)}, t, t') \wedge \\
&\quad \textit{HOLDS}(\text{Possess(me,beer)}, t, t')
\end{aligned}$$

Suppose we have the dsa function appearing in Figure 2 and that the temporal coefficient function falls off linearly from 1.0 at 6:00pm to 0.0 at 10:00pm, so $\textsf{ct}(t) = 1 - t/4$, where t is measured in hours past 6:00pm. Now suppose we have two plans $\mathcal{P}_1$ and $\mathcal{P}_2$ such that for plan $\mathcal{P}_1$

$$P_{now}(\exists t, t'\ (t \leq t' < 6{:}00) \wedge \\ \overline{HOLDS}(\text{Loc(me,home)}, t, t')) \\ \geq \alpha$$

and for plan $\mathcal{P}_2$

$$P_{now}(\forall t, t'\ (t \leq t' < 9{:}00) \rightarrow \\ \neg \overline{HOLDS}(\text{Loc(me,home)}, t, t')) \\ \geq \beta$$

$\mathcal{P}_1$ is preferred to $\mathcal{P}_2$ if

$$\alpha > \frac{1 - .75\beta}{.4}$$

So if $\beta = .9$ then for any α greater than .8125 plan $\mathcal{P}_1$ will dominate $\mathcal{P}_2$, and plan $\mathcal{P}_2$ can be eliminated from consideration.

8 Multiple Goals, Residual Utility, and Computational Issues

The previous sections established bounds on the probabilities of outcomes that could be used to determine whether one plan was preferred to another. These analyses were performed on the utility functions for individual goals, implicitly assuming that utility for other goals and for resource consumption were the same.

The assumption that global utility is linear additive in the goal utilities (Section 4) makes explicit the tradeoff between satisfying the top-level goals, and between satisfying a goal and consuming resources. The assumption that resource consumption (counted in the residual utility term) is independent of the goal utilities means that we can regard resource consumption as a goal as well. Take the two-goal case, for example, for which we can write the equation for the expected utility of a plan $\mathcal{P}$ as follows:

$$\mathbf{EU}(\mathcal{P}) = k_1\mathbf{EU}_1(\mathcal{P}) + k_2\mathbf{EU}_2(\mathcal{P}) + k_r\mathbf{EU}_r(\mathcal{P}) \quad (4)$$

where

$$\mathbf{EU}_i(\mathcal{P}) = \Sigma_c k_i \mathrm{UG}_i(c)\mathbf{P}(c|\mathcal{P}).$$

Now suppose that we have already generated a plan $\mathcal{P}_1$ and we know its expected utility $\mathbf{EU}(\mathcal{P}_1) = u_1$.

Further suppose that we have begun generating an alternative plan $\mathcal{P}_2$, in particular we have generated a subplan that achieves the first goal. From this subplan we can compute

1. an upper bound on the utility associated with first goal, $\mathbf{EU}_1(\mathcal{P}_2) \leq u_{12}$, and
2. a minimum level of resource consumption which provides us with an upper bound on residual utility, $\mathbf{EU}_r(\mathcal{P}_2) \leq u_{r2}$.

We can then calculate that for $\mathcal{P}_2$ to be preferred to $\mathcal{P}_1$ it must *at least* satisfy goal two to the degree

$$\mathbf{EU}_2(\mathcal{P}_2) \geq \frac{u_1 - k_1 u_{12} - k_r u_{r2}}{k_2}$$

which represents the required utility level for the second goal assuming no additional resource consumption. Examining the symbolic structure of the second goal may indicate exactly what propositions must be made true or what quantity level must be attained, and by when, in order to attain that level of utility. The ratio between k_r and k_2 along with the residual utility function indicates how efficiently the second goal must be satisfied as well.

8.1 Plan generation and refinement

The results in previous sections all involved comparing two complete plans; the relationships we provided reduced the question of which was preferable in the sense of maximizing utility to one of establishing a relationship between probabilities over the symbolic attributes that comprise one of the goal expressions. The algorithm in [Hanks, 1990b] exploits both the symbolic content of the relationship and the numeric threshold to limit inference in establishing whether or not this relationship holds. Therefore deciding which of two partial plans is preferable may be considerably cheaper than computing the expected utility of each.

The result earlier in this section demonstrates that given one complete plan and a partial plan we can characterize the interesting possible completions of the partial plan—the ones that satisfy the remaining goals with a certain probability and with a certain effectiveness if the second plan is to be preferred to the first. Therefore deciding if a partial plan is worth pursuing may be considerably cheaper than examining all of its completions.

The question arises in general, however, as to what extent these relationships can aid the planning process. What can we say about the relationship between two partial plans? We are limited in what we can say about partial plans because the typical planning operations used to complete the plans (*e.g.* adding or replacing steps) may have an arbitrary effect on the plan's expected utility. Therefore it will hard to be guarantee that partial plan $\mathcal{P}_1$ is preferable to partial plan $\mathcal{P}_2$ regardless of how both of them are completed.

One special case of partial planning that is amenable to our analysis is that in which the planner's only operation is to *refine* its current plan, which amounts to replacing an abstract action in the plan with a more specific version of that action. More precisely a refinement operator can never increase the set of possible outcomes consistent with the plan's execution, and thus tends to resolve uncertainty about its quality. An estimate of an abstract plan's expected utility might be represented as an interval representing the possible EU values that its completions might take. Refining the plan would tend to narrow the interval. Comparing two abstract plans could then stop as soon as their

expected-utility intervals no longer overlapped[5], and the width of the intervals for the components of the utility function could point to areas of the plan that would benefit from further refinement.

Although it is probably unrealistic to expect any planner to employ only refinement operators, the idea of modeling an execution system using a hierarchy of abstract actions is consistent with Firby's [1989] RAP system, and the architecture for planning and execution proposed in [Hanks and Firby, 1990]. We are currently using that framework to pursue an implementation of the ideas presented in this paper.

9 Summary and Related Work

Our goal in this work was to take intuitions about the nature of goals as they have been used in symbolic planning systems and recast the intuitions in a form in which they might be exploited by decision-theoretic planning algorithms. Our framework involves building a utility model first by identifying the agent's top-level goals plus the residual attributes that measure resource consumption and production in service of those goals. The assumption of utility independence among these attributes means that their interactions can be summarized by $n + 1$ numeric parameters representing the relative weights for the n goals and residual attributes.

For each of the goals one supplies a utility function that divides into two components: the atemporal degree of satisfaction and the temporal weighting coefficient. The former defines what it means to satisfy the goal, either partially or fully. The latter indicates how utility declines as a function of missing the deadline. We demonstrated forms for the degree of satisfaction function given some common goal types: making a symbolic expression true and maximizing a quantity.

We then showed how the model's information, both numeric and symbolic, could be exploited to compare plans: to decide whether one plan's expected utility was greater than another and to generate bounds on the quality of a partial plan in order that it should be chosen over an alternative.

A discussion of related work should begin with a mention of multiattribute decision theory, especially [Keeney and Raiffa, 1976]. What we have done is built a multiattribute utility model for goal-oriented planning problems that feature partial goal satisfaction and deadlines.

Our discussion of strictly ordered goals was motivated by the work in *goal programming* [Schniederjans, 1984] a mathematical optimization technique that deals with ordered conflicting goals.

In the AI literature the work closest to our own is [Wellman and Doyle, 1991], which also analyzes the relationship between goals and preference structures. That paper confronts the question of what it means to say that an agent has some goal. The most fundamental difference between their work and ours is that they begin by examining an agent's preference structure directly and produce a definition of what it means to say that an agent has a goal γ. In contrast, we adopt various intuitive notions about goals at the outset (*e.g.* that they are utility independent at the top level), and structure the agent's utility function (and therefore his preferences) to accommodate those ideas. Our work is mainly oriented toward using the resulting structure to build and exploit representations for concepts like partial satisfaction and temporal deadlines in the process of building and comparing plans.

Etzioni [1991] confronts the problem of decision-analytic *control*: the decision of whether an agent should plan further or act immediately in a given situation. Etzioni's model admits both partial satisfaction of the goal and the idea that the value of achieving the goal will tend to change over time. Both of these elements are supplied directly to the model, in the form of three functions:

- a function $i(g)$ measuring the "intrinsic value" of goal g,
- a function $d(s, g)$ measuring the extent to which goal g is satisfied in state s, and
- a function $F(i(g)d(s, g), s)$ measuring the extent to which the benefit of goal g should be realized in state s.

The first two functions correspond roughly to our atemporal component, the third to our temporal weighting coefficient. Etzioni makes the same assumption we do about the independence of top-level goals. Etzioni's effort is mainly complementary to our own: we provide specific forms for the functions i, d, and F, which he assumes as inputs. We further show how to exploit those functions to make effective plan choices (which he does not), but we do not explore the relationship between building and executing plans (which he does).

10 Conclusion

Our goal is to establish a relationship between decision-theoretic choice and symbolic planning, exploiting the former's normative model of choice and the latter's algorithms for plan generation and refinement.

This paper examined the relationship between utility functions and goals, introducing the concept of a dead-

[5] Or when the intervals narrowed to the point where the distinction between the two did not warrant further attention, as [Russell and Wefald, 1991] point out.

line goal and two accounts of what it means to satisfy one partially.

Doing so required us to explain how to combine utilities for individual goals into a single utility function and how to combine expressions describing the extent of satisfaction for a single goal's temporal and atemporal components into a single measure of goal satisfaction. We used this representation to demonstrate a number of relationships between goal satisfaction and utility maximization, and further showed how these relationships might be used to focus the process of plan generation, projection, and refinement.

References

[Etzioni, 1991] Oren Etzioni. Embedding decision-analytic control in a learning architecture. *Artificial Intelligence*, 1-3(49):129-160, 1991.

[Firby, 1989] R. James Firby. Adaptive execution in complex dynamic worlds. Technical Report 672, Yale University, Department of Computer Science, January 1989.

[Haddawy and Hanks, 1990] Peter Haddawy and Steve Hanks. Issues in decision-theoretic planning: Symbolic goals and numeric utilities. In *DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control.* Morgan Kaufmann, November 1990.

[Haddawy, 1991a] P. Haddawy. *Representing Plans Under Uncertainty: A Logic of Time, Chance, and Action.* PhD thesis, University of Illinois, 1991. (Report no. UIUCDCS-R-91-1719).

[Haddawy, 1991b] P. Haddawy. A temporal probability logic for representing actions. In Richard Fikes James Allen and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 196-207. Morgan Kaufman, San Mateo, CA, 1991.

[Hanks and Firby, 1990] Steve Hanks and R. James Firby. Issues and architectures for planning and execution. In *DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control.* Morgan Kaufmann, November 1990.

[Hanks, 1990a] Steven Hanks. Practical temporal projection. In *Proceedings AAAI*, 1990.

[Hanks, 1990b] Steven Hanks. Projecting plans for uncertain worlds. Technical Report 756, Yale University, Department of Computer Science, January 1990.

[Keeney and Raiffa, 1976] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* John Wiley & Sons, 1976.

[McDermott, 1982] Drew McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101-155, 1982.

[Russell and Wefald, 1991] Stuart J. Russell and Eric H. Wefald. Principles of metareasoning. *Artificial Intelligence*, 1-3(49):361-396, 1991.

[Schniederjans, 1984] Marc J. Schniederjans. *Linear Goal Programming.* Petrocelli Books, 1984.

[Wellman and Doyle, 1991] Michael P. Wellman and Jon Doyle. Preferential semantics for goals. In *Proceedings AAAI*, 1991.

Total Order *vs.* Partial Order Planning: Factors Influencing Performance

Steven Minton **Mark Drummond** **John L. Bresina** **Andrew B. Philips**

Federal Systems Group, Sterling Software Inc.
NASA Ames Research Center
Mail Stop 269-2
Moffett Field, CA 94035

Abstract

In this paper we compare the utility of representing plans as total orders and partial orders, analyzing factors that influence relative planner performance. Most previous work has assumed that representing plans as partial orders is better than representing plans as total orders. Our goal has been to more accurately characterize the conditions under which partial-order planning *is* actually superior. Two main results are presented in this paper. First, we show that the potential benefits of partial-order planning may be retained when using highly expressive planning languages, even though the cost of plan extension may be intractable in the worst case. Specifically, we illustrate how a conservative operator-ordering strategy can guarantee tractability at the expense of introducing unnecessary orderings. Second, we study how solution density, solution clustering, search strategy, and heuristics affect the relative performance of our total-order and partial-order planners.

1 Introduction

For many years, the superiority of non-linear planners over linear planners has been tacitly assumed by the planning community. In a previous paper [15] we examined this intuition, specifically focusing on one aspect of non-linear planning: the use of *partially ordered* plans rather than *totally ordered* plans (see also [8]). We argued that the only significant difference between partial-order planners and total-order planners is planning efficiency. We also pointed out that, in fact, partial-order planning can be less efficient since the search space of partially ordered plans can be larger than that of totally ordered plans. However, we presented a partial-order planner whose search space is guaranteed to be no larger, and is possibly exponentially smaller, than a corresponding search space of totally ordered plans. The key to this guarantee is that the planner's search space is restricted to partially ordered plans that are *unambiguous*; that is, plans in which each precondition is either necessarily true or necessarily false. In addition, we showed that for a propositional planning language, the time cost per node for our partial-order planner is not significantly worse than the cost for a corresponding total-order planner (both are polynomial). Since our partial-order planner can search an exponentially smaller space than the total-order planner but still requires only polynomial time per node, its overall planning efficiency can be considerably greater.

In this paper, we describe two extensions to our previous work. For more expressive planning languages, it is known that the cost of extending partially ordered plans can be intractable [4]. This suggests the following question: does the theoretical advantage of planning with unambiguous partially ordered plans disappear with the introduction of more expressive languages? We demonstrate here that the advantage can be retained for unambiguous planners. Second, we study some factors impacting the relative utility of partial order planning that have not been previously recognized in the literature. Specifically, we examine how the relative performance of partial-order and total-order planning can be affected by the choice of search strategy and heuristic. We study the expected performance relationship in detail for a particular example domain, identifying how general characteristics of the search space impact the performance of our two planners.

2 Background: A Tale of Two Planners

This section reviews relevant terminology and theoretical results established in our previous work [15]. To begin with, we assume that a library of operators is available, where each operator has preconditions, deleted conditions, and added conditions; each

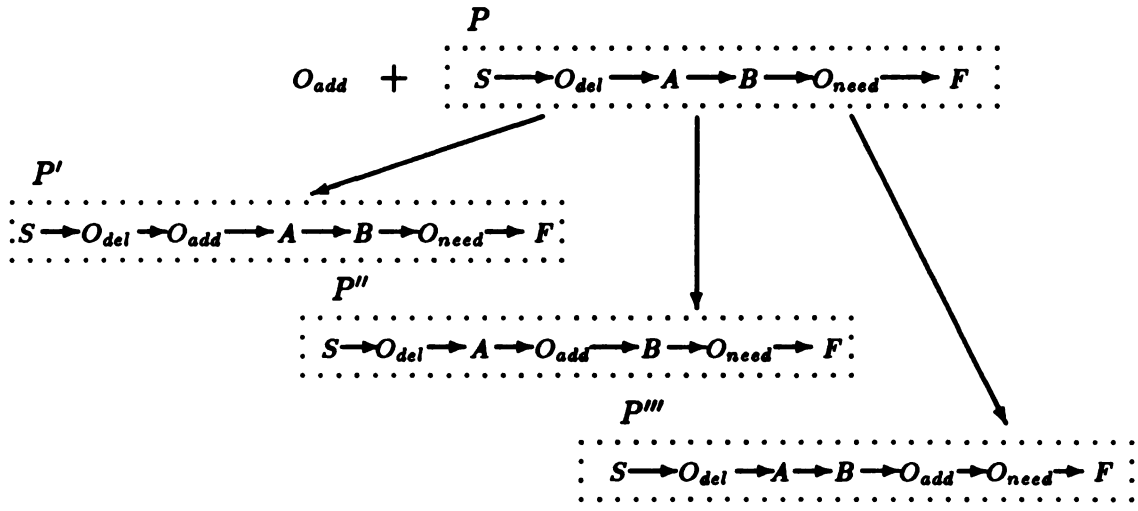


Figure 2: How TO extends a plan: O_{add} is added to plan P to generate three alternatives.

TO(P, G)

1. **Termination:** If G is empty, report success and stop.
2. **Goal selection:** Let c = **select-goal**(G), and let O_{need} be the plan step for which c is a precondition.
3. **Operator selection:** Let O_{add} be an operator in the library that adds c. If there is no such O_{add}, then terminate and report failure. *Backtrack point: all such operators must be considered for completeness.*
4. **Ordering selection:** Let O_{del} be the last deleter of c. Insert O_{add} somewhere between O_{del} and O_{need}, call the resulting plan P'. *Backtrack point: all such positions must be considered for completeness.*
5. **Update goals:** Let G' be the set of preconditions in P' that are false.
6. **Recursive invocation:** TO(P', G').

Figure 1: The TO Planning Algorithm

deleted condition must be a precondition. Each condition must be a non-negated propositional literal. A *plan* consists of an ordered set of instantiated operators, called steps. For convenience, a *planning problem* is represented as a two-step *initial plan*, where the first step asserts a set of initial conditions and the final step has as its preconditions the user-supplied goals. The planning process starts with this initial plan and searches through a space of possible plans. A successful search terminates with a *solution* plan, *i.e.*, a plan in which all steps' preconditions are necessarily true. The search space can be characterized as a tree, where each node corresponds to a plan and each arc corresponds to a plan refinement. A *total-order planner* searches through a space of totally ordered plans; a *partial-order planner* is defined analogously.

Our first planning algorithm, TO, shown in Figure 1, is a total-order planner. TO accepts an unfinished plan, P, and a goal set, G, containing the preconditions of steps in P which are false. The TO procedure first checks whether the goal set G is empty. If so, the plan is finished and the procedure terminates; if not, a goal is selected from G. The function **select-goal** can be any deterministic function that returns a member of G. TO then selects an operator to achieve the goal and inserts the operator into the plan, ordering it with respect to existing operators. The new operator can be inserted anywhere between the point of requirement and the operator which is the "last deleter". Specifically, as used in step 4, the *last deleter* of a precondition c for a step O_{need} is defined as follows. Step O_{del} is the last deleter of c if O_{del} deletes c, O_{del} is before O_{need}, and there is no other deleter of c between O_{del} and O_{need}. In the case that no step before O_{need} deletes c, the first step is considered to be the last deleter.

Figure 2 illustrates TO's plan extension process. This example assumes that steps A and B do not add or delete c. There are three possible insertion points for O_{add} in plan P, yielding the three alternative extensions, P', P'', and P'''.

The pseudo-code in Figure 1 characterizes TO's search space by defining a depth-first procedure for enumerating possible plans. Both planners described in this section can also be implemented as breadth-first procedures, and in that case, both are provably complete [16].

Our second planner, UA, shown in Figure 4, is similar to TO in the way it selects goals and operators; note, however, that UA is a partial-order planner. UA only orders plan steps based on "interactions". Two steps

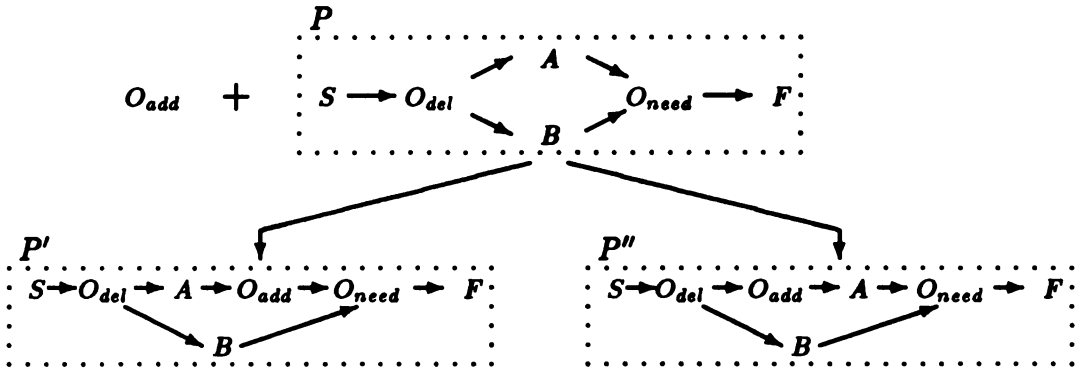


Figure 3: How UA extends a plan: O_{add} is added to plan P to generate two alternatives. The example assumes that O_{add} interacts with step A.

UA(P,G)

1. **Termination:** If G is empty, report success and stop.
2. **Goal selection:** Let c = **select-goal**(G), and let O_{need} be the plan step for which c is a precondition.
3. **Operator selection:** Let O_{add} be an operator in the library that adds c. If there is no such O_{add}, then terminate and report failure. *Backtrack point: all such operators must be considered for completeness.*
4. **Ordering selection:** Let O_{del} be the last deleter of c. Order O_{add} after O_{del} and before O_{need}. Repeat until there are no interactions:
   - o Select a step O_{int} that interacts with O_{add}.
   - o Order O_{int} either before or after O_{add}.

   Backtrack point: both orderings must be considered for completeness.

   Let P' be the resulting plan.
5. **Update goals:** Let G' be the set of preconditions in P' that are necessarily false.
6. **Recursive invocation:** UA(P',G').

Figure 4: The UA Planning Algorithm

in a plan are said to *interact* if they are unordered with respect to each other and there exists a precondition c of one step that is added or deleted by the other.[1] The significant difference between UA and TO lies in step 4: TO orders the new step with respect to *all* others, whereas UA adds *only* those orderings that are required to eliminate interactions. It is in this sense that UA is *less committed* than TO.

Figure 3 illustrates UA's plan extension process. As in Figure 2, we assume that steps A and B do not add or delete c; however, step A and O_{add} interact with respect to some other condition. This interaction yields two alternative plan extensions: one in which O_{add} is ordered before A and one in which O_{add} is ordered after A.

Since UA orders all steps which interact, the plans in its search space are unambiguous: each precondition in a plan is either necessarily true or necessarily false. To see this, consider that a plan is ambiguous only when a precondition c is true in some linearizations and false in others. This can only occur if there are two operators that are unordered with respect to each other and one operator has precondition c and the other adds or deletes c. However, in this case, the operators interact, and they will be ordered by UA. Thus, since UA starts with an unambiguous plan and each recursive call to UA preserves this property, all plans in UA's search space are unambiguous.

Because UA generates only unambiguous plans, there is a tight correspondence between the two planners' search spaces, as illustrated by Figure 5. Each plan in UA's search tree corresponds to a non-empty set of plans in TO's search tree, one TO plan for each linearization of the UA plan. Furthermore, given any two UA plans, the two corresponding sets of TO plans are disjoint. Together, these two properties guarantee that TO's search tree is *at least* as large as UA's search tree.

Formally, let $tree_{ua}$ be the complete search tree for UA on a given problem, let $tree_{to}$ be the complete search tree for TO on the same problem, and let *parent* be a function from a plan to its parent plan in a search tree. Then we can define a mapping $\mathcal{L}$ from plans in $tree_{ua}$ to sets of plans in $tree_{to}$ as follows. Let U be a plan in $tree_{ua}$ and T be a plan in $tree_{to}$; then $T \in \mathcal{L}(U)$ if and only if T is a linearization of U and either both U and T are root nodes of their respective search trees or $parent(T) \in \mathcal{L}(parent(U))$. $\mathcal{L}$ satisfies the following two properties.

[1] As defined in [15], a step that deletes c interacts with one that adds or deletes c because we require that every condition deleted by a step must also be a precondition.

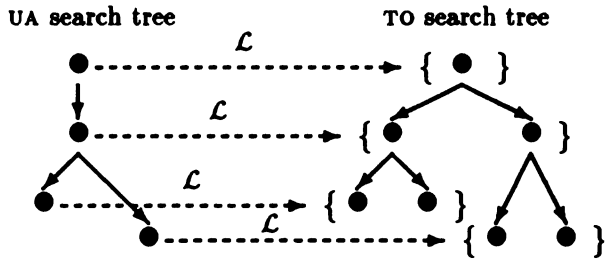


Figure 5: How $\mathcal{L}$ maps from $tree_{ua}$ to $tree_{to}$

1. **Totality property:** For every plan U in $tree_{ua}$, there exists a non-empty set $\{T_1, \ldots, T_m\}$ of plans in $tree_{to}$ such that $\mathcal{L}(U) = \{T_1, \ldots, T_m\}$.
2. **Disjointness property:** $\mathcal{L}$ maps distinct plans in $tree_{ua}$ to disjoint sets of plans in $tree_{to}$; that is, if $U_1, U_2 \in tree_{ua}$ and $U_1 \neq U_2$, then $\mathcal{L}(U_1) \cap \mathcal{L}(U_2) = \{\}$.

Intuitively, $\mathcal{L}$ maps a plan U in $tree_{ua}$ to all linearizations in $tree_{to}$ which share a common derivation ancestry. $\mathcal{L}$ induces a partition on the nodes of $tree_{to}$; consequently, $\mathcal{L}$ enables us to prove that for any given problem, $tree_{to}$ has at least as many nodes as $tree_{ua}$ [15].

How much larger is $tree_{to}$ than $tree_{ua}$? The mapping described above provides an answer. For each plan U in $tree_{ua}$ there are $\mid \mathcal{L}(U) \mid$ distinct plans in $tree_{to}$, where $\mid \mathcal{L}(U) \mid$ is the number of linearizations of U. The exact number depends on how unordered U is. A completely unordered plan has a factorial number of linearizations and a totally ordered plan has only a single linearization. Thus, the only time that the size of $tree_{ua}$ equals the size of $tree_{to}$ is when every plan in $tree_{ua}$ is totally ordered; otherwise, $tree_{ua}$ is strictly smaller than $tree_{to}$, and possibly exponentially smaller.

The relative efficiency of UA and TO also depends on the time required for each planner to visit a node in its search tree. In our previous work we showed that for TO the amortized time cost per node in $tree_{to}$ is $O(n)$, where n is the number of steps in the plan at that node. For UA, the amortized cost per node is only slightly higher; specifically, the cost is $O(e)$, where e is the number of edges in the graph representing the partially ordered plan at that node. Notice that $n \leq e < n^2$. (Throughout this paper, the term "cost per node" will refer to the amortized time cost per node.)

3 How More Expressive Languages Impact Performance

The primary advantage that UA has over TO is that UA's search tree may be exponentially smaller than TO's search tree; UA only pays a small (polynomial) extra cost per node for this advantage. However, thus far we have assumed a very restricted planning language in which the operators are propositional. Most practical problems demand operators with variables, conditional effects, or conditional preconditions. With a more expressive planning language, will the time cost per node be significantly greater for UA than for TO? One might think so, since the work required to identify interacting steps can increase with the expressiveness of the operator language used [7, 10]. If the cost of detecting step interaction is high enough, the savings that UA enjoys due to its reduced search space will be outweighed by the additional expense incurred at each node.

Consider the case for simple breadth-first search. (In the next section, we discuss the implications of using a more sophisticated search strategy.) In [15] we formalized an overall efficiency comparison of UA and TO, assuming a propositional language. Relevant aspects of this comparison are repeated here; later in this section, we consider how our comparison is affected by more expressive operator languages.

Let U be a plan in $tree_{ua}$, and denote the number of steps in U by n_u; the number of edges in U is denoted by e_u. Then for each plan U that UA generates, UA incurs time cost $O(e_u)$, whereas, TO incurs time cost $O(n_u) \cdot \mid \mathcal{L}(U) \mid$, where $\mid \mathcal{L}(U) \mid$ is the number of linearizations of U. Therefore, the ratio of the total time costs of TO and UA is as follows, where $bf(tree_{ua})$ denotes the subtree considered by UA under breadth-first search.

$$\frac{\text{cost}(\text{TO}_{bf})}{\text{cost}(\text{UA}_{bf})} = \frac{\sum_{U \in bf(tree_{ua})} O(n_u) \cdot \mid \mathcal{L}(U) \mid}{\sum_{U \in bf(tree_{ua})} O(e_u)}$$

This cost comparison is specific to the simple propositional operator language used so far, but the basic idea is more general. UA will generally outperform TO whenever its cost per node is less than the cost per node for TO multiplied by the number of TO nodes that correspond under $\mathcal{L}$. Thus, UA can incur an exponential cost per node and still outperform TO in some cases. This can happen, for example, if the exponential number of linearizations of a UA partial order is greater than the exponential cost per node for UA. In general, however, we would like to avoid the case where UA pays an exponential cost per node and, instead, consider an approach that can guarantee that the cost per node for UA remains $O(e)$. The cost per node for UA is dominated by the cost of updating the goal set (step 5) and the cost of selecting the orderings (step 4). Updating the goal set will remain $O(e)$ as long as the plans are unambiguous; thus, it is only the cost of maintaining the unambiguous property (*i.e.*, step 4) that is impacted by more expressive languages. Our approach for efficiently maintaining this property relies on a "conservative" ordering strategy in which opera-

tors are ordered if they even *possibly* interact.

As an illustration of this general approach, consider a simple propositional language with conditional effects, such as "if p, then add q". Thus, an operator can add or delete propositions depending on the state in which it is applied. We refer to conditions such as "p" in our example as *dependency conditions*. (Note that, like preconditions, dependency conditions are also simple propositions.) Chapman [4] showed that with this type of language it is NP-hard to decide whether a precondition is true in a partially ordered plan. For the special case of *unambiguous* plans, this decision can be accomplished in polynomial time. Specifically, since each precondition in an unambiguous plan is either *necessarily* true or *necessarily* false, we can determine the truth value of a given precondition by examining its truth value in an arbitrary linearization of the plan.

In fact, this implies that updating the goal set G, in step 5 of UA, can be accomplished in $O(e)$ time simply by examining a single linearization of the plan. The only other step of UA possibly affected by the additional complexity of the language is step 4, where orderings are introduced to remove interactions. However, UA can resolve interactions in $O(e)$ time if we modify the definition of step interaction as follows: two steps in a plan are said to *interact* if they are unordered with respect to each other and there exists a precondition or dependency condition of one step that can be added or deleted by the other step. So, for example, two steps interact if one step conditionally adds q and the other has precondition q. Note that the first step need not actually add q in the plan, and ordering the two operators might be unnecessary. In general, our definition of interaction is a sufficient criterion for guaranteeing that the resulting plans are unambiguous, but it is not a necessary criterion. Nevertheless, this conservative approach allows interacting steps to be detected via a simple syntactic test that does not increase the time cost of the ordering procedure (step 4).

Apart from the new definition of step interaction, the UA algorithm remains essentially the same.[2] Since TO builds totally ordered plans, it can also handle this language without any significant modification and without any increase in the time cost per node. Thus, UA and TO have exactly the same time cost per node for this new language as they did for the original language, $O(e)$ and $O(n)$, respectively. Moreover, the same relationship holds between the two planners' search spaces; namely, $tree_{ua}$ is never larger than $tree_{to}$ and can be exponentially smaller.

[2]We note that to maintain completeness, TO and UA must be slightly modified so that dependency conditions can also become subgoals [18, 5]. This modification is straightforward (e.g., [14]) and is largely irrelevant to the discussion here.

This example illustrates that the theoretical advantages that UA has over TO can be preserved for a more expressive language. The same "trick" can be used for other languages provided that the UA and TO algorithms remain essentially the same; specifically, both algorithms must consist of the five steps outlined earlier. (For example, in [16] we do this for a language where operators can have variables in their pre- and post-conditions.) To see this, note that for a given UA plan and a corresponding TO plan, steps 1,2, and 3 of the UA algorithm cost the same as the corresponding steps of the TO algorithm. Step 5 of the UA algorithm can be accomplished with an arbitrary linearization of the plan, in which case it will cost at most $O(e)$ more than step 5 of the TO algorithm. Thus, the only possibility for additional cost is in step 4. In general, if we can devise a "local" criterion for interaction such that the resulting plan is guaranteed to be unambiguous, then step 4 can be accomplished in $O(e)$ time.[3] By "local", we mean a criterion that only considers operator pairs to determine interactions; *i.e.*, it must not examine the rest of the plan.

In general, UA can be expected to outperform TO provided that the cost of extending unambiguous, partially ordered plans is not significantly greater than that of extending totally ordered plans. However, the addition of unnecessary orderings can sometimes increase the size of UA's search tree.[4] The magnitude of this increase depends on the specific language, domain, and problem being considered. Nevertheless, we can guarantee that UA's search tree will never be larger than TO's.

The larger lesson here is that the cost of plan extension is not solely dependent on the expressiveness of the operator language, it also depends on the nature of the plans that the planner considers. So, although the extension of partially ordered plans is NP-hard for languages with conditional effects, if the planner is restricted to unambiguous plans, then this worst-case situation is avoided.

4 How Search Strategy Impacts Performance

UA's search space *can be* exponentially smaller than TO's search space, but what does this mean in prac-

[3]As in our previous paper, we assume that the size of the operators and the operator library is fixed for a given domain. So, for example, a local procedure for determining whether two operators interact will take constant time.

[4]It might appear that unnecessary orderings are disadvantageous in that they artificially limit the execution flexibility of the solution plan. However, using a post-processing step whose time complexity is polynomial in the size of the solution plan, it is possible to remove unnecessary orderings, producing a "maximally" unordered plan.

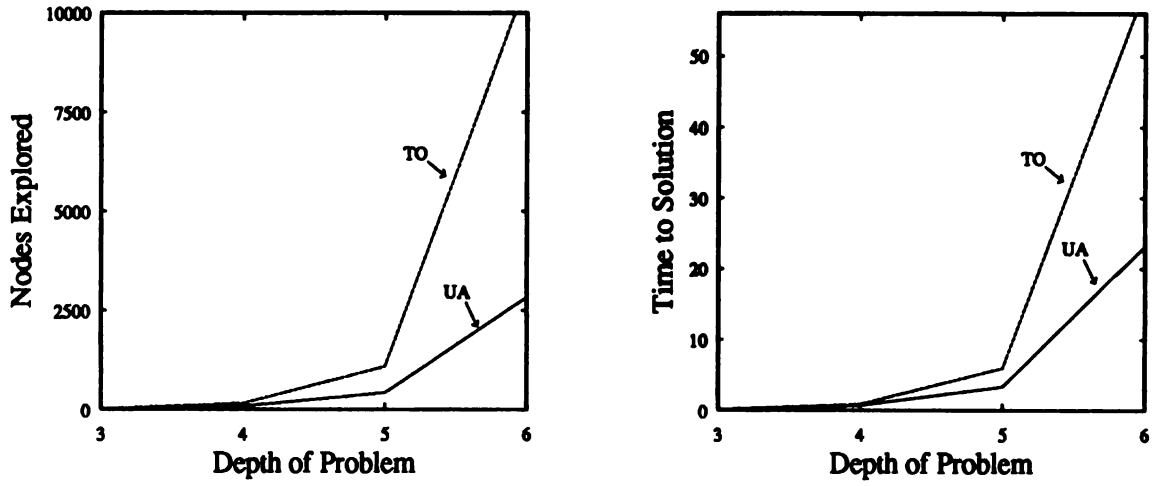


Figure 6: UA and TO Performance Comparison under Depth-First Search

tice? The actual performance of a given planner depends not only on the size of its search space, but also on the particular search strategy used.

For breadth-first search, the analysis is simple[15]. After completing the search to a particular depth, both planners will have explored their entire trees up to that depth. Both UA and TO find a solution at the same depth due to the correspondence between their search trees.[5] Thus, the degree to which UA will outperform TO, under breadth-first, depends solely on the "expansion factor" under $\mathcal{L}$, *i.e.*, on the number of linearizations of UA's plans.

Depth-first search is generally more practical than breadth-first search, but is more difficult to analyze. Notice that for the case of breadth-first search, the two planners are *synchronized* after completely exploring each level, in the sense that for each plan explored by UA, all and only the corresponding plans are explored by TO. However, under depth-first search, UA and TO are not guaranteed to explore corresponding subtrees, since choices made by the two planners are not necessarily synchronized. Specifically, it may be the case that the search trees are enumerated in such a way that corresponding plans are not considered in the same order. In fact, since the planners terminate after finding a solution, it may be the case that the two planners end up exploring entirely different sets of plans (except for the root plan).

Empirically, we have observed that UA does tend to outperform TO under depth-first search, as illustrated by the experimental results in Figure 6. The first graph compares the mean number of *nodes* explored by UA and TO on 44 randomly generated blocksworld problems; the second graph compares the mean planning *time* for UA and TO on the same problems and demonstrates that the extra time cost per node for UA is relatively insignificant. The problems are partitioned into 4 sets of 11 problems each, according to minimal solution "length" (*i.e.*, the number of steps in the plan). For each problem both planners were given a depth-limit equal to the length of the shortest solution.[6] Since the planners make nondeterministic choices, 25 trials were conducted for each problem.

There are several plausible explanations for the observed dominance of UA. As we have pointed out, UA's search tree is never larger than TO's search tree, and in fact, in conducting the above experiments we observed that it was typically much smaller. However, search tree size alone is not sufficient to explain UA's dominance. In particular, the density and distribution of solutions can affect the relative performance of UA and TO under depth-first search.

The solution density of a search tree is the proportion of nodes that are solutions.[7] If the solution density for TO's search tree is greater than that for UA's search tree, then TO might outperform UA under depth-first

[5] For perspicuity, we ignore the fact that the number of nodes explored by the two planners on the last level may differ, assuming that the planners stop when they reach the first solution.

[6] Since the depth-limit is equal to the length of the shortest solution, an iterative deepening [12] approach would yield similar results. Additionally, we note that increasing the depth-limit past the depth of the shortest solution does not significantly change the outcome of these experiments.

[7] This definition of solution density is ill-defined for infinite trees, but we assume that a depth-bound is always provided, so only a finite subtree is explicitly enumerated.

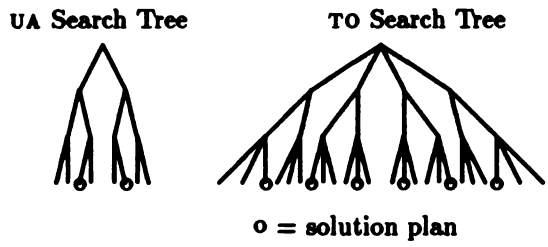


Figure 7: Uniform solution distribution, with solution density 0.25

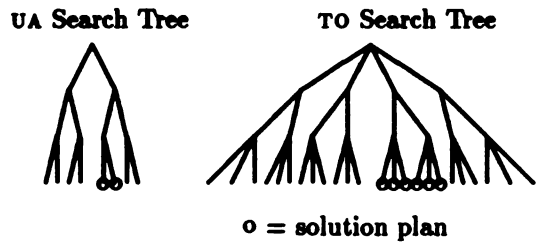


Figure 8: Non-uniform solution distribution, with solution density 0.25

search *even though* TO's search tree is actually larger. For example, it might be the case that all UA solution plans are completely unordered and that all failed plans are totally ordered. In this case, each UA solution plan corresponds to an exponential number of TO solution plans, and each UA failed plan corresponds to a single TO failed plan. The converse is also possible: the solution density of UA's search tree might be greater than that of TO's search tree, thus favoring UA over TO under depth-first search. For example, there might be a single totally ordered solution plan in UA's search tree and a large number of highly unordered failed plans. Since each UA failed plan corresponds to a large number of TO failed plans, the solution density for TO would be considerably lower.

For our blocksworld domain, experiments indicate that the observed dominance of UA under depth-first search is *not* due to UA having a greater solution density than TO. We compared the solution densities for UA and TO using a randomized search strategy, a type of Las Vegas algorithm[3], that we refer to as "iterative sampling" (*cf.* [13]). The iterative sampling strategy explores randomly chosen paths in the search tree until a solution is found. To select a path, the system traverses the tree from the root to a leaf, choosing randomly at each branch point. If the leaf is a solution then search terminates; if not, the system returns to the root and selects another path. The same path may be examined more than once, since no memory is maintained between successive trials. Iterative sampling can be used to estimate the solution density in a tree, assuming that the tree's branching factor is relatively uniform.

In our experiments we found that when both UA and TO use iterative sampling, they expand approximately the same number of nodes on our set of blocksworld problems.[8] This result suggests that the density of solutions in the two trees is not significantly different. This is not surprising, since every linearization of a UA solution plan is a solution plan for TO, and similarly for failed plans. Furthermore, at least in the blocksworld, there is no *a priori* reason to suppose that the number of linearizations for a solution plan should be greater than the number of linearizations for a failed plan. Thus, for this domain, it seems reasonable to suppose that the two planners' solution densities are similar.[9] Of course, we are not suggesting that this is true for all domains; indeed, it is straightforward to design domains where the two planners' solution densities differ significantly.

Since solution density is insufficient to explain UA's dominance under depth-first search, we need to look elsewhere for an explanation. We hypothesize that the *distribution* of solutions provides an explanation. We note that if the solution plans are uniformly distributed among the leaves of the search tree, then both planners can be expected to search the same number of nodes. This is illustrated by the search tree schematic in Figure 7. Consequently, to explain the observed dominance of UA over TO the solutions must *not* be uniformly distributed, that is, solutions must tend to cluster. This hypothesis makes sense, since for many types of problems, a decision near the top of the search tree can lead to an entire subtree of failed plans, as illustrated in Figure 8.

Our experiments with iterative sampling are consistent with this explanation. The fact that there is no difference between UA and TO under iterative sampling, but that there *is* a difference under depth-first search suggests that solutions are non-uniformly distributed. To see this, suppose that $tree_{ua}$ has a given solution density and non-uniform distribution. We expect $tree_{to}$ to be larger than $tree_{ua}$ but to have a similar solution density and a distribution that is no more uniform. Consequently, depth-first search will perform relatively poorly on $tree_{to}$ since there is a greater cost associated with each incorrect choice. On the other hand, iterative sampling should perform similarly on both trees, since it is much less sensitive to the distribution of solutions.

Although our blocksworld domain may be atypical, we

[8] The iterative sampling strategy was depth-limited in exactly the same way that our depth-first strategy was. We note, however, that the performance of iterative sampling is relatively insensitive to the actual depth-limit used.

[9] Other factors could conceivably explain our results, for example, the trees might have highly non-uniform branching factors. However, this seems unlikely.

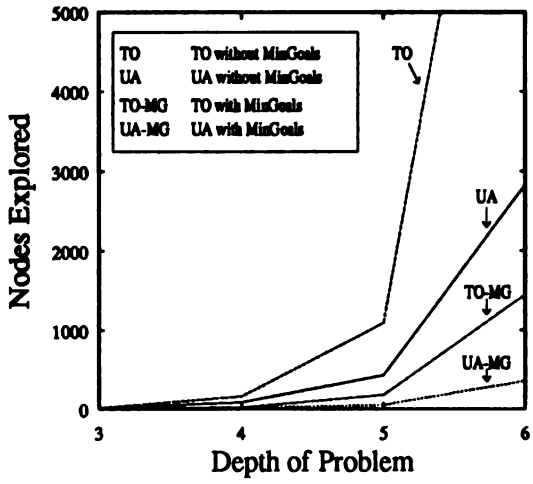


Figure 9: Depth first search with and without min-goals

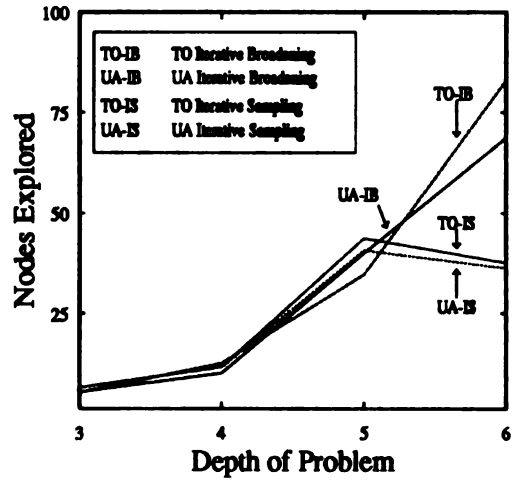


Figure 10: Iterative sampling & iterative broadening, both with min-goals

conjecture that our results are of general relevance. Specifically, for distribution-sensitive search strategies like depth-first search, one can expect that UA will tend to outperform TO. For distribution-insensitive strategies, such as iterative sampling, non-uniform distributions will have no effect. We note that while iterative sampling is a rather simplistic strategy, there are more sophisticated search strategies, such as iterative broadening[9], that are also relatively distribution insensitive.

5 The Relative Importance of Heuristics

In the previous sections we investigated the advantage that UA has over TO for the case of uninformed search strategies. Our previous work [15] has also suggested that the utility of partial-order planning might increase with the use of certain types of heuristics. In order to investigate this, and to gauge the relative importance of heuristics, we evaluated the performance of UA and TO under a standard planning heuristic discussed in our previous work.

The "min-goals" heuristic selects amongst alternative plan extensions by preferring the extension with the fewest false preconditions (*i.e.*, fewest goals). Figure 9 shows the impact of this heuristic on the behavior of UA and TO under depth-first search. This result is consistent with our observations in the previous section, where UA outperforms TO under depth-first search. Although the heuristic biases the way the two planners' search spaces are explored, it appears that its effect is largely independent of the partial-order/total-order distinction, since both planners are improved similarly on a percentage basis. For example, under depth-first search on the problems with solutions at depth six, UA improved 88% and TO improved 87%. Thus, we find no evidence that UA exploits this heuristic more effectively than does TO, although in [15] we pointed out that this was theoretically possible. (We note that a planner which is less committed than UA might better realize this possibility.)

Notice also that TO *with* min-goals outperforms UA *without* min-goals. Although the effectiveness of min-goals is domain dependent, we find it interesting that, in these experiments, the use of min-goals makes more difference than the use of partial-orders. After all, the blocksworld domain originally helped motivate the development of partial-order planning and most subsequent planning systems have employed partial-orders. While not deeply surprising, this result does help reinforce what we already know: more attention must be paid to specific planning heuristics such as min-goals.

The effect of combining iterative sampling and min-goals is shown in Figure 10. Under this strategy, UA and TO perform equally well, consistent with the behavior described in the previous section. The figure also shows the impact of min-goals on iterative broadening [9]. This more sophisticated search strategy initially behaves like iterative sampling and evolves into depth-first search as the breadth-cutoff increases [13]. Iterative broadening is distribution-insensitive in its early stages, but as the breadth-cutoff increases, it becomes more sensitive to solution distribution. However, it is not surprising that UA and TO perform similarly under iterative broadening in our experiments,

since solutions were found very early on.

The performance of both UA and TO under either search strategy in Figure 10 is significantly better than under depth-first search with min-goals as shown in Figure 9. (Note that the two graphs in Figures 9 and 10 have very different scales.) One reason for this difference is that when min-goals mistakenly favors a node not on a solution path, depth-first search explores the entire subtree below that node (*cf.* [17]). In contrast, if either of the two iterative techniques makes a mistake early in the search, the cost of the mistake is relatively small.

There is a caveat to our experimental results. For simplicity, we implemented iterative sampling in such a way that it is not complete when used with min-goals. (In fact, we removed a problem from our randomly generated set because iterative sampling with min-goals would never terminate.) In our implementation, only those plan extensions with the fewest goals are considered at each choice point; however, the strategy can be made asymptotically complete by using the heuristic to probabilistically bias the search, rather than using it to prune alternatives. In any event, our results clearly illustrate the utility of this heuristic in our blocksworld domain, since on 43 of our 44 problems, a solution exists in the subspace preferred by the heuristic.

6 Related Work

McAllester and Rosenblitt [6] define a systematic nonlinear planner, SNLP, that is very similar to UA in that it introduces orderings until a plan is unambiguous. However, their motivation is quite different. UA and TO were designed to facilitate a comparative analysis of total-order and partial-order planning. SNLP, on the other hand, was designed to illustrate the concept of *systematic* search, i.e., the same plan is never considered more than once. Unlike SNLP, neither UA nor TO maintain an explicit causal link structure and, therefore, are not guaranteed to be systematic. It is unknown whether the advantage of guaranteeing systematicity is worth the extra cost of maintaining a causal link structure for each plan. As Kambhampati [11] points out, systematicity is not guaranteed to have a direct correlation with the efficiency of planning.

Barrett, Soderland, and Weld [1, 2] build on McAllester and Rosenblitt's work, comparing a version of the partial-order SNLP planner with two total-order planners derived from it. In particular, they examine how partial-order and total-order planning compare for problems with independent, serializable, and non-serializable goals. Our work is complimentary since we examine a different set of issues, namely, how solution density and distribution, together with search strategy, affect the relative performance of partial-order and total-order planning.

7 Conclusion

While we have considered only two relatively simple planners, our study has examined issues of general relevance and has achieved two main results.

First, we argued that the potential benefits of partial-order planning may be retained even with highly expressive planning languages. Although plan extension may be intractable in the worst case, conservative operator-ordering strategies can guarantee that plan extension remains tractable at the expense of introducing unnecessary orderings. Specifically, we showed that for a language with conditional effects, the cost of UA's plan extension remains $O(e)$, yet its search space may still be exponentially smaller than that of TO.

This paper has also studied the fundamental utility of partial orders as a knowledge representation framework. Most previous work has assumed that representing plans as partial orders is better than representing plans as total orders. Our goal has been to better characterize the conditions under which partial-order planning *is* actually superior. Our work is the first to study how solution density, solution clustering, search strategy, and heuristics affect the relative performance of planners based on total-orders and partial-orders. We have conjectured that the relative advantages of an unambiguous partial-order planner over a corresponding total-order planner depend critically on a number of factors. For example, our experiments suggest that distribution-sensitive search strategies, such as depth-first search, can benefit more from partial orders than can search strategies that are distribution-insensitive.

Our observations regarding the interplay between plan representation and search strategy raise new concerns for comparative analyses of planners. Historically, it has been assumed that representing plans as partial orders is simply "better" than representing plans as total orders. The results presented in this paper begin to tell a more accurate story. As we have pointed out, there are a variety of subtle factors that can affect the relative advantages of a particular planning style.

References

[1] A. Barrett, S. Soderland, and D.S. Weld. The effect of step-order representations on planning. Technical Report 91-05-06, Univ. of Washington, Computer Science Dept., 1991.

[2] A. Barrett and D.S. Weld. Partial-order planning. Technical Report 92-05-01, Univ. of Washington, Computer Science Dept., 1992.

[3] G. Brassard and P. Bratley. *Algorithmics - Theory and Practice.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[4] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 1987.

[5] G. Collins and L. Pryor. Achieving the functionality of filter conditions in a partial order planner. In *AAAI Proceedings*, 1991.

[6] McAllester D. and D. Rosenblitt. Systematic nonlinear planning. In *AAAI Proceedings*, 1991.

[7] T. Dean and M. Boddy. Reasoning about partially ordered events. *Artificial Intelligence*, 1988.

[8] M. Drummond and K.W. Currie. Goal-ordering in partially ordered plans. In *IJCAI Proceedings*, 1989.

[9] M.L. Ginsberg and W.D. Harvey. Iterative broadening. In *AAAI Proceedings*, 1990.

[10] J. Hertzberg and A. Horz. Towards a theory of conflict detection and resolution in nonlinear plans. In *IJCAI Proceedings*, 1989.

[11] S. Kambhampati. Multi-contributor causal structures for planning: A formalization and evaluation. Technical Report 92-018, Arizona State University, Dept. of Computer Science and Engineering., 1992.

[12] R.E. Korf. Depth-first iterative deepening: An optimal admissible tree search. *Artificial Intelligence*, 1985.

[13] P. Langley. Systematic and nonsystematic search strategies. In *Proceedings of AI Planning Systems*, 1992.

[14] S. Minton. *Learning Search Control Knowledge: An Explanation-based Approach.* Kluwer Academic Publishers, Boston, Massachusetts, 1988. Also available as Carnegie-Mellon CS Tech. Report CMU-CS-88-133.

[15] S. Minton, J. Bresina, and M. Drummond. Commitment strategies in planning: A comparative analysis. In *Proceedings IJCAI*, 1991.

[16] S. Minton, J. Bresina, M. Drummond, and A. Philips. An analysis of commitment strategies in planning: The details. Technical Report 91-08, NASA Ames, AI Research Branch, 1991.

[17] J. Pearl. *Heuristics: Intelligent Search Strategies For Computer Problem Solving.* Addison-Wesley Publishing Company, Inc., 1984.

[18] E.P.D. Pednault. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence*, 4(4):356–372, 1988.

[19] E.D. Sacerdoti. The nonlinear nature of plans. In *Proceedings IJCAI*, 1975.

A Reactive Planner that Uses Explanation Closure

Andrew R. Haas
Department of Computer Science
The University at Albany
1400 Washington Avenue
Albany, New York 12222

1 Introduction

Imagine a robot that accepts commands in English. Suppose a user says "Pick up the book behind you". The robot turns around and sees a book. It must infer that when the user spoke, this book was in the same place it is now. Since the robot was facing the other way at that time, the book was behind the robot, and therefore it is the book the user referred to. The inference goes something like this: "The only thing that's happened here since the user spoke is my turning around. Turning around wouldn't move the book, so when the user spoke the book was in the same place it's in now". Notice that the robot is not trying to predict the future; it is making an inference about a past situation, one that it could not observe at the time. Notice also that the inference is in two parts: first find the relevant events, then show that none of these events could have moved the book.

The second part of this inference demands a solution for the frame problem: the problem of showing that a given action will *not* change a given fluent, without explicitly listing all the fluents the action does not change (Hayes 1986). Most proposals for the frame problem rely on a default rule: anything that isn't known to change will remain the same. Haas (1987) and Schubert (1990) have argued that persistence should not be inferred entirely by default; we need axioms stating that certain actions do not change certain fluents. However, that does not mean that for each action we list every fluent that it does not change. Instead we list, for each fluent, the set of actions which *can* change that fluent. The axiom that lists these actions is called an ***explanation closure*** axiom. If an event e is not on the list of events that can change the fluent, we infer that it will not change the fluent. Davis (1990) seems to have discovered the same idea independently.

We divided the problem of inferring persistence into two parts: finding the relevant events, and showing that they do not change the fluent. AI workers usually concentrate on the second part of the problem. For example, they assume that a plan is given and they try to show that the actions in the plan will not change a given fluent. In the world of everyday life, finding the relevant events can be hard. Some of the events might be actions of other agents, and predicting them will take us into deep waters. I will consider a simple case: the robot is the only agent who can move things from place to place, and the robot does not need to make inferences about the future, only the past. If the robot's actions are the only relevant events, and the robot is reasoning about the past, then the robot can remember all the relevant events. However, this is not enough. In order to construct a list of all relevant events in an interval I, the robot must not only remember the events; it must infer that there were no other relevant events. Suppose that $[A_1, ..., A_n]$ is a list of all actions that the robot remembers performing during interval I. The robot must be able to infer that since it always remembers the actions it performs, the list $[A_1, ..., A_n]$ contains all the actions it performed during interval I. This is an example of ***autoepistemic reasoning***: the robot knows that its knowledge of its own past actions is complete, so it infers that the actions it knows about are the only ones that happened. Moore (1985) recognized the importance of autoepistemic reasoning, and proposed an autoepistemic logic to capture such reasoning. However, I will argue that Moore's system does not capture the autoepistemic reasoning needed in this case. I propose instead that the robot's observation of its own knowledge is a process like perception. Having observed its own knowledge, the robot can make inferences from this observation using (once again) ordinary monotonic logic.

I have implemented these ideas by building a simulated robot, which accepts commands in English and executes them. The robot uses reactive planning (Agre and Chapman 1987; Chapman 1991). It looks at its surroundings, chooses its next action on the basis of what it sees, executes the action, and then looks again. The motivation for reactive planning is that predicting the effects of actions is very hard. Agre and Chapman wrote programs that played video games, and it was hard to predict the future because the games included enemies who acted in unpredictable ways. My robot

does not have this problem. Its environment includes only one other agent, the user, and the user has a very limited ability to change the environment. The robot still has great difficulty in predicting the effects of its actions, but for a different reason: because of its ignorance of the present state of its environment. The robot does not start with a map of its environment; it learns about the environment by perception. Given this limited knowledge, the robot cannot create detailed plans for the future. Suppose I go to the library and I bring the robot along to help me look for books. We come in the door and I tell the robot, "Get me a copy of Herstein's *Topics in Algebra*". The robot has never been in this library before, and it cannot be sure that the library even has a copy of Herstein. It cannot find a list of actions and predict that when it has finished executing those actions in the order given, it will have a copy of Herstein in its hand. But it *can* think of a sensible thing to do first. Looking around for a card catalog would be a good choice. If the robot doesn't try to plan far ahead, but merely decides what to do next, it is using reactive planning. This is the why the robot does not need to predict the future. However, it still has to reason about the past, as we saw in the "Pick up the book behind you" example.

2 Reactive Planning and Logic

Our robot does reactive planning in first-order logic. It does not prove that by executing plan P, it can achieve goal G. Instead it proves that by executing an action A, it can get closer to achieving goal G. Advocates of reactive planning seldom worry about exactly what "closer" means, but if we are serious about logic we need to specify the meanings of our predicates. Let us consider exactly what the robot proves before it acts, and why we expect the robot to achieve its goal.

Our robot has a set of *commands*: instructions that it can send to its effectors. Executing a command will result in some kind of action (perhaps unsuccessful), and the execution will be completed in a finite time. We can describe the result of executing a command by the relation $\mathrm{r}(c, s_1, s_2)$, which means that if the world is in state s_1 and the robot executes the command c, it is possible that when the execution is complete, the world will be in state s_2. The execution of a command always produces some result, though it may not be useful; so for every c and s_1, there exists s_2 such that $\mathrm{r}(c, s_1, s_2)$. Notice we are not assuming that only the robot's actions can change the world. That would be a very strange assumption for a reactive planner. We are assuming that the new state depends on both the initial state and the command that the robot executes. It may be that the differences between the new and old states are largely due to forces outside the robot's control (Schubert 1990).

A goal is just a set of states: the states in which the goal is achieved. For each goal there is a set of states in which the goal is not achieved, but it is still possible for the robot to achieve the goal. We say that the goal is *accessible* from these states. Some of these states are closer to the goal than others, so for each goal g there is a partial order on states. We write $s_2 <_g s_1$, which means the goal g is accessible from states s_1 and s_2, but in s_2 the robot is closer to achieving the goal than in s_1. Suppose the goal g is to reach a certain point P. For each state s we might define $d(s)$ as the length of the shortest obstacle-free path from the robot's present location to P. We could then say that $s_2 <_g s_1$ holds iff $d(s_2) < d(s_1)$.

Let g be a goal and s_0 a state from which the goal is accessible. Consider a sequence of states $s_0, s_1, s_2, \ldots$ such that $s_{i+1} <_g s_i$ for $i \geq 0$. That is, each state is closer to goal g than the one before. We postulate that there must exist an n such that the goal is achieved in state s_n. In other words: if the robot keeps getting closer to the goal, eventually it gets there. Suppose that at time t the world is in state s_1, and executing command c will produce a state that is closer to the goal g. That is, for every s_2 such that $\mathrm{r}(c, s_1, s_2)$, we have $s_2 <_g s_1$. Then we say that executing c at time t will bring the robot closer to goal g.

Suppose the robot is given a goal g at time t_1, and the world is in a state s_1 from which g is accessible. The robot tries to find a command c such that executing c in s_1 will bring the robot closer to goal g. If it finds such a command, it will execute the command at time t_1. The execution will be complete at some time t_2, and the world will be in a new state s_2. The robot then goes back to the top of the loop: it tries to find a command c such that executing c in s_2 will bring the robot closer to goal g. According to our postulates, the robot will eventually reach a state s_n in which the goal is achieved.

On this account, it is clear how one would produce a formal proof of correctness for a reactive planner. First define the effects of the commands by the relation $\mathrm{r}(c, s_1, s_2)$; then for each goal g, define the relation $s_2 <_g s_1$ and show that it satisfies the postulates; then show that the robot always chooses a command that brings it closer to the goal. This account is of course simplified. We have not allowed for the time spent in planning, and we have limited our goals to predicates on single states. However, this is enough to make the point: logical analysis applies to reactive planners just as well as other planners. It is possible to give a precise analysis of the notion of getting closer to a goal, and to prove that a reactive planner will eventually reach its goal.

3 The Robot's Domain

The robot's environment is inside a building. The rooms contain tables and books, and the doorways

may be labeled with numbers. The space in each room is divided into squares like a checkerboard, and the robot moves one square at a time. The user gives commands by typing English and pointing with a mouse. For example, one may type "Put this book here". The words appear above an image of the room the robot is currently in, seen from above. The user draws an arrow from the word "this" to a book, and from the word "here" to a point in the room.

The robot has an internal clock, whose reading is a rational number. I will use the phrase "time N" to mean "the time when the robot's clock reads N", with the understanding that N is a number. The robot uses two coordinate systems. It has an *egocentric* coordinate system: a Cartesian coordinate system whose origin is always the present location of the robot. The sensors report the locations of objects using egocentric coordinates. The robot also has a fixed coordinate system. When the user starts the robot, the internal clock reads 0.0, and the robot chooses its present location as the origin of the fixed coordinate system. Since space is merely a checkerboard, the robot can rely heavily on dead reckoning. It has only to remember its own actions since time 0.0, and do some arithmetic, in order to compute its present location in the fixed coordinate system. In a realistic robot, dead reckoning is still important, but its accuracy is limited. The robot might use dead reckoning to approximate its present position in a fixed coordinate system, and then match nearby objects against known objects to get a more accurate location (Davis 1990).

Suppose there is a book in front of the robot, 3 squares to the right and 4 squares ahead. The robot's sensors will report this with two atomic sentences: *book*(*book*1) and *ec*(*book*1, 3, 4, 87.5). The second and third arguments of *ec* are egocentric coordinates of the book. The last argument gives the time of the observation. The constant *book*1 will reappear in all future reports about the same book. Such a constant is called a *sensor name*. If the sensors always use the same sensor name to denote the same object, then the robot can always recognize the books (and also the tables). Vere's Basic Agent also has the ability to recognize all the objects it encounters (Vere 1991). This is clearly unrealistic: books and tables often look much alike, and the problem of showing that this book *here* is the same one the robot saw before should be left to the reasoner, not solved automatically by the sensors. However, given that the robot can always convert egocentric coordinates to fixed coordinates, and objects don't move unless the robot moves them, it would be fairly easy for the robot to identify objects by their location. It must find the fixed coordinates of the object, and then check for a known object that ought to be at those coordinates. I have implemented this treatment for one type of object (doors), and it would be easy to extend it to other objects.

The robot's actions are simple:

- advance - advance one square.
- turn(D) - turn and face in direction D. D is left, right, or behind.
- grasp(X,Y) - grasp the object at egocentric coordinates (X,Y). (X,Y) must be a neighboring square (that is, $-1 \leq X \leq 1, 0 \leq Y \leq 1$).
- release - release the object you are grasping.

The user cannot move objects, so the robot can be certain that if it puts an object somewhere, the object will stay put. The same thing is often true for human beings in the world of everyday life. If I put a book on my desk, and come back an hour later, I can be pretty sure of finding it where I left it. This is not just good fortune. We expend a lot of effort to make sure that objects will stay put, because planning would be very difficult if they didn't. We set up barriers: stout walls and doors with locks. We also set up social conventions: if I have a habit of wandering into other people's offices and taking things, I am likely to end up in jail. So the "stay-put" property is not just an arbitrary simplification of the robot's domain. It holds in large parts of the everyday world, and it is essential to human planning.

Since the robot plans to acquire knowledge, it must understand how its own sensors operate. The robot can see any object that is in the same room, has an egocentric Y coordinate ≥ 0, and is not concealed by a wall. All corners are right angles; aside from this constraint, the shape of a room can be arbitrarily complex. If the robot is told "Get the book on the table in room 5", it may have to perform an elaborate search of room 5 to find the table. In general, when the robot sees an object it perceives all the properties of that object. There are two exceptions: to read the title of a book or the number on a doorway, the robot must be directly in front of the book or the doorway. So if the robot is looking for a book with a given title, it must travel around the room to inspect each book.

The program that drives the robot is written in Prolog. One can of course use Prolog to implement any kind of reasoning, but this program uses it in the most direct way: the robot's knowledge consists of definite clauses, and the robot uses Prolog's depth-first, left-to-right theorem-proving algorithm. Most of the code is declarative: each clause can be read as a sentence of first-order logic, and the robot's conclusions follow from these sentences. There are minor extensions to Prolog's logic: for example, the robot uses the familiar device of proving $(P \rightarrow Q)$ by asserting P, proving Q, and then retracting P.

When the user enters a command, the robot parses it and builds a semantic representation, which it uses as a goal for planning. The representation includes

definite and indefinite descriptions. Finding the referents of these descriptions involves planning to acquire knowledge. Immediately after parsing the input, the robot turns completely around, so that it can see the arrows the user has drawn from words to places in the room. This gives the knowledge it needs to identify the referents of descriptions like "this book".

The robot now enters its planning and acting loop. The arguments of this loop include the goal and the time T when the robot completed its last action. It first tries to use information acquired after the last action to identify the referents of descriptions. Next it tries to prove that the goal has been achieved. If it fails to prove this, it tries to find an action which, if executed at time T, will bring it closer to the goal. Strictly speaking the action will not be executed at time T, because the planning process itself takes time. Since nothing moves until the robot acts, this makes no difference. Having chosen an action, the robot executes that action and returns to the top of the loop.

Suppose the user says "Put this book on this table". The robot identifies the book and the table and goes and gets the book. Now it must find the egocentric coordinates of the table, so that it can carry the book to the table. Suppose the robot discovers that the table is no longer visible because the robot has turned its back. It can infer the egocentric coordinates by remembering the last time it saw the table, remembering the actions it has performed since that time, and predicting how those actions will change the egocentric coordinates of the table.

4 Introspection

The robot cannot always observe the effects of its own past actions, so it needs to infer them. The first thing it must do is to find out what actions it performed and when. Given times T1 and T2, the robot wants to form a list of all triples f(A,T3,T4) such that the robot executed A from time T3 to time T4, and the interval (T3,T4) overlaps the interval (T1,T2). If the robot performs an action A from time T3 to T4, it will record the sentence exec(b, n_3, n_4), where the term b is the robot's name for the action A, and n_3 and n_4 are numerals denoting the numbers T3 and T4. That is, the robot remembers its actions. Let n_1 be the numeral denoting the number T1, and n_2 the numeral denoting the number T2. The robot can form the desired list of triples by executing the following Prolog goal at T2 (or later):

```
1
findall(f(A,T3,T4),
        (exec(A,T3,T4),overlap(T3,T4,n1,n2)),
        L)
```

Notice that this works only if two conditions are fulfilled: the robot executes the "findall" after T2, and the last two arguments of "overlap" are numerals. If the robot executes the "findall" at some time T0 before T2, the resulting list will not contain any actions between T0 and T2 – because the robot does not know about its own actions before it executes them. The robot concludes that it will not perform any actions between T0 and T2 – which may well be false. Now suppose the last two arguments of "overlap" are not numerals. For example, the robot's language might contain the term internal(H,M), which denotes the reading of the robot's internal clock when the time is M minutes past hour H (I omit the date for brevity). If the robot's internal clock reads 729.5, and the robot knows that internal(12,15) = 729.5, then the robot knows that it is now 12:15. If the robot wants to find out what actions it performed between 8:15 and 9:15, it might execute the following:

```
2
findall(f(A,T3,T4),
        (exec(A,T3,T4),
         overlap(T3,T4,internal(8,15),internal(9,15)),
        L)
```

Suppose the robot does not know the numerical values of the terms internal(8,15) and internal(9,15) – in other words, it has not synchronized its internal clock with the clock on the wall. Then the "overlap" goal will fail, producing an empty list of triples. The robot concludes that it performed no actions between 8:15 and 9:15 – which may well be false.

With careful programming, we can ensure that goal (1) is always called after time T2, and the last two arguments of "overlap" are always numerals. But the program will be fragile – careless modification might easily lead to calling (1) at the wrong time, or with the wrong arguments, and deriving a false conclusion. We can avoid this fragility by making the assumptions explicit in the program, so that violating them will lead to failure instead of false conclusions. We do this by adding a new source of beliefs: introspection. Perception tells the robot about events in the outside world, while introspection tells the robot about events in its own head. In particular, suppose the robot finds all provable instances of a goal G. Introspection will create a belief that describes the goal G, the time when the robot searched for provable instances, and the result. Given this information, we can explicitly check that the search for actions between T1 and T2 occurs after T2, and the last two arguments of "overlap" in goal G are numerals.

In order to represent "I found all provable instances of goal G at time T", the robot's representation language must include a term that denotes G. Since G is a wff of the robot's representation language, the language must include names for its own wffs. That is, it must include quotation. The robot uses a quotation device due to McCarthy (1979). For each variable x in the language

there is a constant `$x$, and in the intended model of the robot's beliefs, the constant `x denotes the variable x. For each n-ary functor f of the language there is an n-ary function letter `$f$. In the intended model of the robot's beliefs, the function letter `f denotes a function that maps n expressions $e_1, ..., e_n$ to the expression $f(e_1, ..., e_n)$. For example, the function letter `¬ denotes a function that maps each wff to its negation, and the function letter `∧ denotes a function that maps two wffs to their conjunction.

It is easy to see that this notation allows us to form a name for every expression of the language, by attaching the symbol ` to each functor of the expression. For example, the term `f(`$x$) denotes the term $f(x)$. We can also include variables under quoted function letters. We can write $(\exists n.numeral(n) \wedge believe(`p(n)))$ to indicate that the robot believes an atomic sentence with predicate letter p and a numeral n as its argument. Suppose we are given a quoted term that denotes another term – say, `f(`a). We can find the term it denotes by removing the quotation marks to get f(a).

Now suppose that at time T the robot finds all provable instances of a goal P, and the result is a list of n instances, Ps_1,...,Ps_n. Having performed this operation it can learn by introspection "I found all provable instances of P at time T, and the instances I found were Ps_1,...,Ps_n". In order to state this information, it is convenient to introduce an introspection predicate similar to Prolog's "findall". Let T be a term, P a goal, L a list, and N a number. search(T,P,L,N) means that:

- At time N the robot searched for all provable instances of goal P.
- It found a list of instances Ps_1,...Ps_n.
- The list L contains the denotations of the terms Ts_1,...,Ts_n.

For example, suppose that at time 50.0 the robot calls the goal

```
3
search(`f(`A,`T3,`T4),
       `exec(`A,`T3,`T4),`overlap(`T3,`T4,`40.0,`50.0)),
       L,
       N)
```

If the only action between time 40.0 and time 50.0 was an advance in the interval from 45.0 to 46.0, the robot will get back L=[f(advance,45.0,46.0)].

Using this predicate we can state the axiom that the robot will use to find a list of all actions it performed between time T1 and T2. Let numeral(N1,T1) mean that N1 is a numeral denoting the number T1. Let actions(L,T1,T2) mean that L is a list of all triples f(A,T3,T4) such that the robot executes A from T3 to T4 and the interval (T3,T4) overlaps the interval (T1,T2). We write

```
4
actions(L,T1,T2) :-
   numeral(N1,T1),
   numeral(N2,T2),
   search(`f(`A,`T3,`T4),
          `exec(`A,`T3,`T4),`overlap(`T3,`T4,N1,N2)),
          L,
          N),
   T2 ≤ N.
```

This axiom says that if the robot finds all provable instances of the goal (exec(A, T3, T4), overlap(T3, T4, n_1, n_2)) at time N, if n_1 and n_2 are numerals, and if time T2 is before N, then the robot will find all the actions it performed between T1 and T2. Suppose the robot tries to use this axiom, and the time N is before T2. The search goal will return a list L that contains no actions between N and T2. The goal T2 ≤ N will fail, and the robot will not draw the (possibly false) conclusion that it will execute no actions between time N and time T2.

Axiom (4) says that under certain conditions, the robot knows all correct solutions for a certain goal. That is, the solutions the robot knows are the only ones that exist. Clark (1978) described a general technique for this kind of reasoning. For each pure Prolog program P, he defined a set of first-order axioms called the *completion* of P. He showed that if the completion is true, then for any goal G in the language of that program, the solutions found by the program are the only ones that exist. Our robot cannot use Clark's technique, because the completion of the robot's program will normally be false. Since the robot's program changes as perception creates new beliefs, we should phrase this more carefully: for most times T in the robot's lifetime, the completion of the robot's program at time T is false. This is true for two distinct reasons. First, the completion of the program says that the actions the robot knows about are the only ones it ever executes. At time T, the robot knows only about the actions it performed before T, so the completion says that the robot will never perform any actions after T. This will be false for most T. Second, the completion entails the sentence $t_1 \neq t_2$, where t_1 and t_2 are any two distinct ground terms in the language of the program. There exists a numeral N such that the sentence internal(12,15)=N is true; but the completion contradicts this, because N and internal(12,15) are two distinct terms. The axiom (4) handles these two problems by explicitly ruling out the troublesome cases. The principle "The solutions I know are the only ones that exist" can be very useful, but it holds only in certain very restricted circumstances. We need a notation that allows us specify these circumstances, on a case-by-case basis. Clark's technique does not allow us to specify the circumstances tightly enough. It does not allow us to say "The predicate "exec" is complete for those cases where its last argument is <

T" or "The predicate "overlap" is complete for those cases where its last two arguments are numerals".

Moore's autoepistemic logic is also inadequate for our purposes, for similar reasons. The belief operator of autoepistemic logic has no time argument, so we cannot say that at time T the robot knows about all actions it performed before T. There is no quantification over terms of the representation language, so we cannot say "The last two arguments of the predicate "overlap" are numerals". In fact most of the work is limited to the propositional case, which is useless for our purposes.

Despite these technical differences, it is interesting to contrast the intuitions behind Moore's work and the present paper. Moore believed that introspection is similar to logical deduction, and he devised an "autoepistemic logic", which combines introspection and deduction in one system. My intuition is that introspection is not much like deduction - it is more like perception of the external world. Certainly introspection resembles deduction in one way: both operate inside the robot's head, requiring no input from the external world. However, introspection also resembles perception in a very important way. As Moore (1985) pointed out, introspection is *indexical*. When the robot uses introspection to discover that it believes P, what it learns is "I believe P now". In the same way, when the robot uses perception to discover that it is standing in a room, what it learns is "I am standing in a room now".

The work on autoepistemic logic includes soundness theorems, which guarantee that if the premisses are true, the conclusions derived by autoepistemic logic are also true. It is easy to check that the beliefs created by introspection are true, so we are not losing precision by doing autoepistemic reasoning without an autoepistemic logic.

5 Explanation Closure

Let us apply explanation closure to the robot's domain. Let $ec(B,X,Y,T)$ mean that (X,Y) are the egocentric coordinates of object B at time T. Let $fc(B,X,Y,T)$ mean that (X,Y) are the fixed coordinates of object B at time T. Suppose the robot knows what its own fixed coordinates were at time T1, and it has a list of all actions that it performed between T1 and time T2. It uses explanation closure to find its own fixed coordinates and direction at T2. An advance action increments (or decrements) the X (or Y) coordinate by 1 - depending on the robot's direction. This action leaves the direction unchanged. The turn action changes the robot's direction and leaves the fixed coordinates unchanged. Any action that is not equal to turn or advance leaves the fixed coordinates and direction unchanged. There is of course no need to *list* the actions that leave the coordinates unchanged. In fact there are two, but if there were a dozen or a hundred it wouldn't matter: the explanation closure axiom lists only the actions that do change the coordinates.

The robot also needs to know which room it is currently in. The robot can recognize doorways, and as it travels around the building it records which room each doorway leads to. Each time it passes through a doorway, it finds out which room it has just entered. If it has never entered this room before, it chooses a new constant to denote the new room. The robot records that at time T it entered room R. Now suppose the robot needs to know what room it is in at time T2. If it entered room R1 at time T1, and it has not entered any room between T1 and T2, it must still be in room R1 at T2. The robot always records the fact that it entered a room, so it can argue as follows: "If I had entered a room between T1 and T2 I would know about it; so if I don't remember entering any room between T1 and T2, I am still in room R1 at T2". The following axioms support this line of reasoning.

```
5
robot_in_room(R1,T2) :-
    enter_room(R1,T1),
    T1 ≤ T2,
    not_enter_room(T1,T2).
6
not_enter_room(T0,T2) :-
    numeral(N0,T0),
    numeral(N2,T2),
    search('T1,
           ('enter_room('R,'T1),'between('T1,N1,N2)),
           [],
           T3),
    T2 ≤ T3.
```

Note that the interval (T0,T2) must be before T3, the time when the robot executes the search.

Suppose that the robot wants to find the fixed coordinates of a book B at time T. Only the robot can move the book, and when the robot moves a book it always sees the book it is moving. Therefore the book will always be where it was the last time the robot saw it. Of course there might be no last time - the robot might want to know where a book was before the first time the robot saw it. Since the robot cannot move the book before seeing it, its location before the robot saw it must have been the same as its location the first time the robot saw it. For example, the user might say "Get the book on the table in room 5". Suppose the robot has never been to room 5, and has never seen the book the user is referring to. When it gets to room 5 it sees a book sitting on the table. It can reason as follows: "Since this is the first time I've seen that book, it must have been on that table when the user spoke. Therefore that is the book the user referred to".

If the robot sees an object X at time T, it records this fact by adding a sentence see(X,T) to its knowledge

base. The second argument of "see" is a numeral denoting the time T. The first argument is a sensor name for the object X. Suppose the robot sees object X at time T1, and T1 ≤ T2. To verify that T1 was the last time before T2 when the robot saw X, the robot must show that it did not see X between T1 and T2. It follows the usual line of reasoning: "If I saw X between T1 and T2, I would know about it; so if I don't remember seeing X between T1 and T2, I did not see it". sensor_name(C) means that C is a sensor name. denotes(T,X) means that the term T denotes the entity X. The program includes an axiom schema for this predicate. If T is a term and QT is the quote name of T, then the sentence

```
7
denotes(QT,T)
```

is an axiom. We can now state the axioms for inferring the fixed coordinates of a book at one time, given the coordinates at another time.

```
8
not_seen(Book,T0,T2) :-
    denotes(C,Book),
    sensor_name(C),
    numeral(N0,T0),
    numeral(N1,T1),
    search(`T1,
           (`see(C,`T1),between(N0,`T1,N2)),
           [],
           T3),
    T2 ≤ T3.
9
fc(Book,X2,Y2,T2) :-
   see(Book,T1),
   T1 ≤ T2,
   not_seen(Book,T1,T2),
   fc(Book,X2,Y2,T1).
10
fc(Book,X1,Y1,T1) :-
   see(Book,T2),
   not_seen(Book,-1.0,T2),
   T1 ≤ T2.
```

The first axiom says that if the robot cannot prove a sentence of the form see(C,T1), where C is a sensor name for the book and T1 is a numeral for a time between T0 and T2, then the robot did not see the book between T0 and T2. The next axiom says that if the robot saw the book at T1, T1 ≤ T2, and the robot did not see the book between T1 and T2, then the book's coordinates at T2 are the same as its coordinates at T1. The final axiom says that if the robot did not see the book at any time before T2, and T1 ≤ T2, then the book's coordinates at T1 are the same as the book's coordinates at T2.

This example shows that in some cases, it is easier to predict the effects of past actions than of future actions. The reason is that the robot knows a good deal about the past from observation. If the robot wants to infer the present location of a book, it does not need to consider its own past actions – it merely reasons that the book is where it was when the robot last observed it. A similar situation arises if the robot wants to know what object (if any) it was grasping at a past time T. After each action the robot scans its surroundings, and if it is grasping an object it will see that it is grasping that object. So the robot can always answer the question "What was I was grasping at time T?" without reasoning about the effects of its own actions. The robot uses inference only as a substitute for perception, and it never tries to infer something that it can find out by looking.

However, closer consideration shows that even when reasoning about the past, the robot cannot always use axioms (8-10) to prove that a book remains in one place. The problem is that sometimes the robot must reason about a book it has never seen – and if it has never seen an object, it has no sensor name for that object. Suppose the user says "Go get the book on the table in room 5". The robot chooses a new constant c to denote the referent of "the book on the table in room 5", and it assumes that the user described this object correctly: it is a book, and is on a table in room 5 at the time when user types his command. The robot sets out for room 5 to find the book. It is using reactive planning, so after each action it considers the situation and decides what to do next. When the robot is halfway to room 5, it must infer that the book c is still in room 5, and therefore it should continue going toward room 5. The robot cannot prove a goal of the form not_seen(c, t_1, t_2), because the constant c is not a sensor name. In fact the robot has no sensor name for the book c until it reaches room 5, so it cannot use axiom (9) or (10) to prove that this book remains where it was when the user typed his command.

The solution is of course to use an explanation closure axiom, which describes the possible ways that a book could move from one room to another. We define the predicate move_nothing_to_new_room(T1,T3), which means that the robot did not carry any object from one room to another in the interval (T1,T3). When the robot enters a room, it records this fact, and when the robot is grasping an object, it can perceive this. Therefore we write the following axiom.

```
11
move_nothing_to_new_room(T1,T3) :-
    numeral(N1,T1),
    numeral(N3,T3),
    search(`T2,
           (`enter_room(`R,`T2),
            `grasping(`X,`T2),
            `between(N1,`T2,N3)),
           [],
           TK),
```

TK $\geq$ T3.

Suppose the robot needs to know the present location of a book. Applying axiom (11) will take time that is at least proportional to the number of times the robot has entered another room between T1 and T3. Applying axiom (9) or (10) will take constant time, assuming certain optimizations given below. Also, note that (11) applies only in the case where the robot has not moved any object to another room in the interval (T1,T3). A more general axiom would allow the robot to form a list $c_1, ..., c_n$ of all objects that it moved to another room in the interval (T1,T3), and then prove that a given object was not moved because it is not a member of that list.

Our treatment of autoepistemic reasoning is *sentential*. The robot describes its own beliefs by describing the sentences that represent them. It does not use (for example) sets of possible worlds. If we already have a program that uses sentences to represent its beliefs, then the sentential approach is the simplest and most obvious. Why introduce new entities called "propositions", if we can do the job with an entity that is already there in the program? This argument aside, it is hard to see how one could reproduce the program's reasoning in a non-sentential theory. In stating axiom (8), it is crucial to distinguish between sensor names and other constants that denote objects – such as the ones the robot uses to name objects described by the user. It is not clear how to represent such a distinction without explicitly talking about terms and sentences.

6 Implementation

The program is written in SICSTUS Prolog, except that the simulated environment and the graphics are in C. The implementation is mostly straightforward, but two issues are worth mentioning.

It seems desirable for the robot's database to contain information that is as recent as possible. Suppose the robot wants to find its own fixed coordinates at time t_1, and the most recent known coordinates are at time $t_0 < t_1$. After finding the coordinates at t_1 the robot will save them in its database. Suppose the robot later needs to find its own fixed coordinates at time $t_2 > t_1$; it can use the known information about t_1, and it need not recompute the effects of the actions it performed between t_0 and t_1. The same technique applies to several other predicates.

Suppose the robot needs to find its own fixed coordinates at t_2, and the last known coordinates were at time t_1. The robot must form a list of all actions it performed between t_1 and t_2. To do this the robot must call "findall", and that takes time proportional to the total number of actions executed so far – even though the length of the resulting list of actions is the number of actions executed between t_1 and t_2. This is intolerable; it means that the longer the robot continues to operate, the slower it will get. My solution to this problem is to ensure that the unit clauses exec(A,T3,T4) appear in the Prolog database in a fixed order: most recent first. I then use a conditional cut (O'Keefe 1990, p. 234) to terminate the search after the first clause exec(A,T3,T4) such that T4 < T1. This ensures that all solutions are found, but the time to execute findall is independent of the total number of actions the robot has ever executed.

This approach yields reasonable speed. Running SICStus version 2.1 on a SPARCstation 1, I recorded the time needed to choose the next action for 184 primitive actions, rounded down to the nearest 1/8 of a second. The environment consists of a hallway with six doors, opening on rooms that contain books, tables, and a clock. Most of the times are under a second, all but one are under two seconds. There is a clear tendency to slow down over time. This is to be expected, since the robot's knowledge base grows as it travels around and learns more about the environment.

Schubert (1990) described an algorithm for explanation closure that combines goal-driven and data-driven inference. He was interested in predicting the effects of a proposed plan (plan tracking, as he called it), rather than inferring the unobserved effects of actions that the robot has already executed. He used a default inference procedure that works roughly as follows: if a condition held at $i < k$, and it is not explicitly marked as changed (or possibly changed) at some point between i and k, it persists until k. Schubert's inference procedure used defaults, but its conclusions are logical consequences of the explanation closure axioms, which are written in ordinary first-order logic. So Schubert's procedure is not a theorem-prover for a new logic – it is a special-case theorem-prover for standard logic.

In order to ensure the soundness of his defaults, Schubert had to make data-driven inferences. He had to be certain that before the algorithm applied default rules at step k, the data-driven inferences had already found every condition that might have changed since i, and marked it – either marked it as "definitely changed", or else marked it as "possibly changed". This allowed him to prove that if a condition was not marked, the explanation closure axioms implied that the condition had not changed between i and k. This in turn meant that the conclusions of the default rules were logical consequences of the explanation closure axioms.

Our procedure also uses a combination of goal-driven and data-driven inference. When the robot goes through a doorway, it finds out what room it has just entered and records the information for later use. If the robot needs to know what room it is now in, it will use axiom (5) to find the last room it entered, and (6) to confirm that it has not entered any other room since. This is similar to Schubert's procedure. The difference is that axiom (6) explicitly says "If the

robot cannot remember entering another room in the interval (T0,T2), then it did not enter another room". In Schubert's work this kind of knowledge exists only in the mind of the programmer – it is not stated in the program's knowledge base.

7 Conclusions

Explanation closure allows us to solve the frame problem without using the nonmonotonic rule "Assume things don't change unless you know they do". Explanation closure offers an alternative to that particular rule – not an alternative to nonmonotonic reasoning in general. It is possible and probably necessary to write the explanation closure axioms themselves in some logic that allows generalizations with exceptions. If we are trying to describe the everyday world, we may find that there are a large number of events that could possibly change a given fluent, but most of them are very unlikely. That is, we may encounter the *qualification problem* (Shoham and McDermott, 1988). We may wish to omit all but the most common causes of change from the explanation closure axiom. Then the explanation closure axiom will have exceptions: sometimes the fluent will change, even when none of the events listed in the explanation closure axiom occurred. To allow for these exceptions, the axiom must be written in something other than first-order logic.

The same point applies to introspection. Our robot uses introspection to infer that an object is still in the last place he saw it. This kind of reasoning is surely common in everyday life, but it is fallible: sometimes an object disappears when you thought you had left it in a safe place. So axiom (9) must allow exceptions, and it too must be written in something other than first-order logic.

Granted that we need a logic for nonmonotonic reasoning, it is far from obvious which problems require that logic. It is commonly assumed that we must solve the frame problem with a nonmonotonic logic, or else give up all hope of a logical solution. Explanation closure gives a simple and efficient solution with ordinary logic. The case of autoepistemic reasoning is more subtle. It is clear that the reasoning is non-monotonic: if I add a sentence P to my beliefs, I can no longer use autoepistemic reasoning to infer that I don't believe P. However, autoepistemic reasoning does not automatically imply autoepistemic logic. We have seen that if we regard the observation of one's own beliefs as something like perception, we can give a precise analysis of autoepistemic reasoning without using a new logic. And this technique is easy to implement.

Acknowledgements

Thanks to Len Schubert for advice and encouragement. This work was supported by National Science Foundation grant number IRI-9006832.

References

Agre, Philip E., and Chapman, David. 1987. Pengi: An Implementation of a Theory of Activity. Proceedings of the Sixth National Conference on Artificial Intelligence,268-272.

Chapman, David. 1991. Vision, Instruction, and Action. Cambridge, Massachusetts: MIT Press.

Clark, K. L. 1978. Negation as Failure. in H. Gallaire and J. Minker, eds., *Logic and Databases.* New York: Plenum Press.

Davis, Ernest. 1990. Representations of Commonsense Knowledge. San Mateo, California: Morgan Kaufmann.

Etherington, David W.; Kraus, Sarit; and Perlis, Donald. 1991. Nonmonotonicity and the Scope of Reasoning. Artificial Intelligence 52:221-261.

Haas, Andrew R. 1987. The Case for Domain-Specific Frame Axioms. in F. M. Brown, ed., *The Frame Problem in Artificial Intelligence.* Morgan Kaufmann.

Hanks, S., and McDermott, D. Default Reasoning, Non-monotonic Logics, and the Frame Problem. in *Proceedings of AAAI-86.*

Hayes, Patrick. 1986. What the Frame Problem Is and Isn't. in Z. Pylyshyn, ed., *The Robot's Dilemma: the Frame Problem in Artificial Intelligence.* Norwood, New Jersey: Ablex Publishing Corporation.

McCarthy, John, and Hayes, Patrick. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. in B. Meltzer and D. Michie, eds., *Machine Intelligence 4.* Edinburgh: Edinburgh University Press.

McCarthy, John. 1979. First-Order Theories of Individual Concepts and Propositions. in B. Meltzer and D. Michie, eds., *Machine Intelligence 9.* New York: Halsted Press: 120-147.

Moore, Robert C. 1985. Semantical Considerations on Non-Monotonic Logic. Artificial Intelligence 25:75-94.

O'Keefe, Richard A. 1990. The Craft of Prolog. Cambridge, Massachusetts: MIT Press.

Schubert, Lenhart. 1990. Monotonic Solution of The Frame Problem in The Situation Calculus: An Efficient Method for Worlds with Fully Specified Actions. in Kyburg, H.E.,Loui,R.,and Carlson, G.,eds., *Knowledge Representation and Defeasible Reasoning.* Dordrecht: Kluwer Academic Publishers: 23-68.

Shoham, Yoav, and McDermott, Drew. 1988. Problems in Formal Temporal Reasoning. Artificial Intelligence 36:49-61.

Vere, Steven. 1991. Organization of the Basic Agent.
SIGART Bulletin 2:164-168.

UCPOP: A Sound, Complete, Partial Order Planner for ADL

J. Scott Penberthy
IBM T.J. Watson Research Center
P.O Box 704
Yorktown Heights, NY 10598
jsp@watson.ibm.com

Daniel S. Weld
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98105
weld@cs.washington.edu

Abstract

We describe the UCPOP partial order planning algorithm which handles a subset of Pednault's ADL action representation. In particular, UCPOP operates with actions that have conditional effects, universally quantified preconditions and effects, and with universally quantified goals. We prove UCPOP is both sound and complete for this representation and describe a practical implementation that succeeds on all of Pednault's and McDermott's examples, including the infamous "Yale Stacking Problem" [McDermott 1991].

1 Introduction

The investigation of techniques for reasoning about actions and plans is split into two camps. One camp has looked at formal characterisations of languages for describing change while another has attempted to build actual planners, often losing a precise understanding of their programs in a forest of pragmatic choices. A few researchers have described complete algorithms rigorously, but all these approaches suffer from one of two liabilities.

- Either the planners only handle the restrictive STRIPS representation (*e.g.*, TWEAK [Chapman 1987] and SNLP [McAllester and Rosenblitt 1991]),
- or the planners represent plans as totally ordered sequences of actions (*e.g.*, Rosenschein's [1981] and Kautz's [1982] bigression planners, and McDermott's [1991] PEDESTAL).

Since consensus suggests that partial order planning is preferable to total order approaches [Minton *et al.* 1991,Barrett *et al.* 1991,Barrett and Weld 1992], we pondered the void in the space of rigorous planners. McDermott [1991] clearly believed our quest doomed: "if you want a completeness theorem for your planner, you had better build a linear planner."

In this paper we show McDermott was overly pessimistic. We describe UCPOP, a Partial Order Planner whose step descriptions include Conditional effects and Universal quantification.[1] Both universal and existential quantification are permitted in the step preconditions, effect preconditions, effect postconditions, and goals. Although UCPOP assumes a closed world (actions cannot add or delete from the fixed universe of objects) and does not allow domain axioms or disjunctive preconditions, it is considerably more expressive than other rigorous partial order planners.

The UCPOP algorithm starts with an initial, dummy plan that consists solely of a "start" step (whose effects encode the initial conditions) and a "goal" step (whose preconditions encode the goals). UCPOP then attempts to complete this initial plan by adding new steps and constraints until all preconditions are guaranteed to be satisfied. The main loop makes two types of choices: supporting "open" preconditions and resolving "threats."

- If UCPOP has not yet satisfied a precondition (*i.e.*, it is "open"), then all step effects that could possibly be constrained to unify with the desired proposition are considered. UCPOP chooses one effect nondeterministically[2] and then adds a causal link [McAllester and Rosenblitt 1991] to the plan to record this choice.
- If a third step (called a "threat") might possibly interfere with the precondition being supported by the causal link, then UCPOP nondeterministically chooses a method to resolve the threat: either by reordering steps in the plan, posting additional subgoals, or by adding new equality constraints.

[1] The name, which we pronounce as "yoo-see-pop", is an anagram of the capitalised letters.

[2] In fact, domain dependent information can be used to guide the choice. Backtracking ensures that all choices will be eventually considered.

Once UCPOP has successfully created and protected a causal link for every goal in the plan, it halts and returns a solution.

1.1 Money at Home, Wisdom at Work

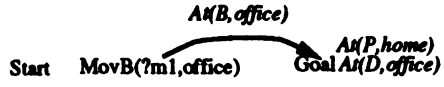
Adopting an example from [Pednault 1988], suppose we had a single briefcase, B, and wanted to use it to move objects. Pednault formalises this simple domain with three operators: `MovB(l)`, for moving a briefcase and its contents, `PutIn(x)` for putting an item x in the briefcase, and `TakeOut(x)` for removing items from the briefcase. Our version of these operators is seen below:

```
TakeOut(x)
  PRECOND :  x ≠ B
  EFFECTS :  ¬In(x)

PutIn(x,l)
  PRECOND :  x ≠ B ∧ At(B,l) ∧ At(x,l)
  EFFECTS :  In(x)

MovB(m,l)
  PRECOND :  At(B,m) ∧ m ≠ l
  EFFECTS :  At(B,l)
             At(z,l) ∀z | In(z) ∧ z ≠ B
             ¬At(B,m)
             ¬At(z,m) ∀z | In(z) ∧ z ≠ B
```

We now demonstrate how UCPOP generates plans with these actions. Since the algorithm is nondeterministic, we assume a convenient order; in practice, some backtracking might be required to find a correct plan. Our example begins with three items: a briefcase B, a paycheck P, and a dictionary D. All three start at home, with the briefcase containing the paycheck. The goal is to have the paycheck at home and the dictionary at the office. The initial plan is depicted as follows:

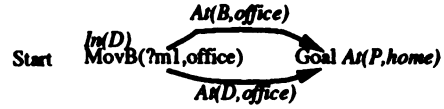
In each diagram we use *italics* to represent open preconditions which are treated as subgoals. When these preconditions are still open (*i.e.*, have not been satisfied), we display them next to the step that requires them. Steps, which are instances of operators, are written in typeface. Arrows between steps denote causal links, showing which subgoals a step has satisfied.

UCPOP selects the goal *At(B,office)*, satisfying it by creating a new `MovB(?m1,office)` step and making a causal link from it to the `Goal` step.[3]

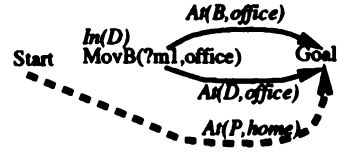


[3] Although this creates another subgoal *At(B,?m1)* we have omitted it and its links for simplicity.

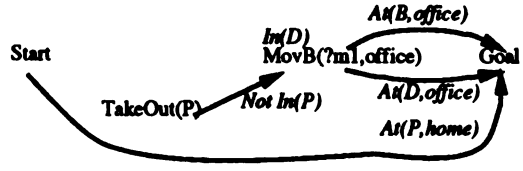
UCPOP then selects the subgoal *At(D,office)*, creating another link from the same step used in the previous goal. This adds a subgoal *In(D)* to the `MovB(?m1,office)` step:



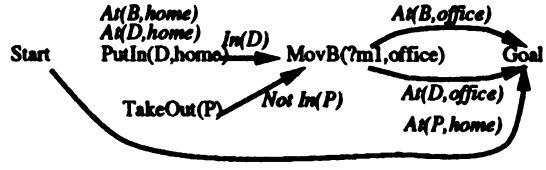
The next goal selected is *At(P,home)*. Since the paycheck is at home to begin with, UCPOP can use this fact to satisfy the final goal as well; UCPOP records this decision by adding a link from `Start` as shown:



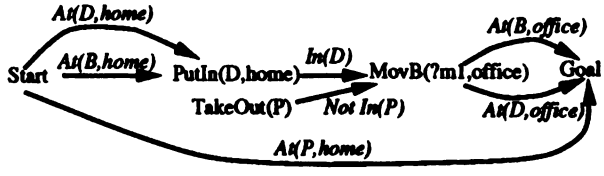
The dashed link denotes a *threat* from the `MovB` step: if the paycheck is left in the briefcase, then moving the briefcase will negate the initial condition, *At(P,home)*, and jeopardise the supported goal. UCPOP eliminates the threat by posting a subgoal $\neg In(P)$ which, in turn, is satisfied by adding `Takeout(P)` to the plan:



UCPOP then selects the subgoal *In(D)* and creates a causal link from another new step `PutIn(D,?m1)`, ensuring that the dictionary is in the briefcase before the briefcase is moved. The new step also generates two more subgoals to ensure that the briefcase and dictionary are spatially coincident:



Finally, UCPOP generates two more links from the initial `Start` step to satisfy the preconditions of the `PutIn` step. This also causes the free variable `?m1` to become bound to *home*. Since no more unsatisfied subgoals are found and since no threats are detected, UCPOP halts with the following partially ordered plan:



	Run Time	Unifies	Plans
Briefcase	510	177	12
U Briefcase	670	160	11
Homeowners	580	48	24
Sussman	690	233	12
U Sussman	1810	1065	20
YSP	2410	1768	20

Figure 1: These data are taken from an implementation of UCPOP on an IBM RS/6000 model 520. Times are in milliseconds.

1.2 Pragmatics and Implementation

The UCPOP algorithm has been implemented in Allegro Common Lisp and runs on a variety of platforms.[4] To maintain completeness we use A* or IDA* search to implement the exploration of nondeterministic choices. Figure 1 shows the actual run times for the briefcase example and others found in the literature [McDermott 1991,Pednault 1986,Pednault 1988,Sussman 1975]. The number of unifications and the total number of partial plans that UCPOP considered when solving a particular planning problem are also shown.

The problems in figure 1 are summarised as follows:

- **Briefcase.** Pednault's conditionalised briefcase domain. The "U" before a problem indicates the use of universal quantification in postconditions, as described in this paper.
- **Homeowners.** You bought a home, turn on the water, and water pours out of holes in the wall. You can paste over the holes and/or fix the plumbing. However, pasting holes before fixing the plumbing is useless [Pednault 1991].
- **Sussman.** We tested on two versions of the Sussman anomaly. The "U Sussman" problem foregoes the *clear*(b) axiom and instead uses $\forall x \neg on(x, b)$, introducing numerous subgoals.
- **YSP.** McDermott's *Yale Stacking Problem*, a variation of the Sussman anomaly that uses the *above* predicate.

We observe that UCPOP found the Yale Stacking Problem quite easy, repudiating McDermott's [1991] belief that it would be "almost impossible for a nonlinear planner. (p 405)" The combinatorics of handling universal preconditions and multiple interacting steps are exemplified by increasing time.

1.3 Overview

In this paper we make the following contributions:

- A clear and simple description of the UCPOP planning algorithm.
- A proof of UCPOP's soundness.
- A proof of UCPOP's completeness.

In the next section we review Pednault's ADL (the language UCPOP uses to represent actions) and detail a few simplifying assumptions. We then (sections 3 and 4) describe a representation for partially ordered plans and present the UCPOP algorithm. Section 5 presents our core results: proofs of soundness and completeness. The paper concludes with a brief discussion of related work.

2 Representing Actions

Frustrated with the restrictive STRIPS representation but frightened by the prospect of implementing a planner for the full situation calculus, we gravitated naturally toward Pednault's [1986,1989] Action Description Language (ADL). ADL is essentially a reformulation of the situation calculus into *action schemata*, akin to the add and delete lists of STRIPS [Fikes and Nilsson 1971]. ADL is more expressive than STRIPS yet less expressive than full, first-order logic.

2.1 Action schemata

The semantics of ADL are based on the algebraic structures (models) used to characterise states of the world. An *action*, a, in ADL is a set of state pairs $\prec s, t \succ$ where action a can be executed in state s to produce state t. The association between a state s and state description ϕ is indicated by the "models" symbol $\models$ and is written $s \models \phi$. An *action schema* in ADL, which more closely resembles the add and delete lists of STRIPS, characterises a set of possible actions. Schemata are described by four optional groups of clauses:

1. Precond, the preconditions,
2. Add and Delete, a set of formulae describing the set of tuples to be added (deleted) to the interpretation of some relations R in the resulting state t, and
3. Update, a set of relations describing how functions change in t from s.

Pednault's version of our `MovB(m,l)` schema is then:

```
MovB(l)
    Add :  At(B,l)
           At(z,l) ∀z | In(z) ∧ z ≠ B
 Delete :  At(B,m)
           At(z,m) ∀z | In(z) ∧ z ≠ B
```

We merge the two Add and Delete categories into a single Effects category. The notion of "adding" a

[4] Send mail to `weld@cs.washington.edu` for information on acquiring the code.

tuple for $R()$ is encoded as before. Deleting a tuple from $R()$ is encoded by asserting $\neg R()$. We note that this technique is merely a syntactic variant of Pednault's action representation. In this paper we also forbid the use of the UPDATE category as well as disjunction in preconditions.[5] We insist that all necessary preconditions be explicitly stated. This diverges from Pednault's [1986] approach, where he assumed preconditions could be inferred from the world state. Free variables in schemata are implicitly existentially quantified, serving as placeholders for future interpretations. Finally, equality constraints can only involve variables or variables and constants. All constants are assumed unique; their value cannot be changed through actions.

2.2 Secondary preconditions

Pednault represents action schemata internally as sets of *causation* and *preservation* preconditions collectively known as "secondary preconditions." Intuitively, a causation precondition of an action A for some relation R, Σ_R^A, specifies all conditions under which A will cause R to be added to the world state. The preservation preconditions of an action A and relation R, Π_R^A completely specify the conditions under which R is *not* deleted from the world state; hence, the conditions that "preserve" the truth value of R. For example, the following is a preservation condition for $At(u,v)$ with respect to action `MovB(m,l)`:

$$\Pi_{At(u,v)}^{MovB(m,l)} \equiv \neg[(u = B \wedge v = m) \vee (u \neq B \wedge v = m \wedge In(u))]$$

The first disjunct above corresponds to moving the briefcase from location `m`, thus affecting $At(B,m)$. The second disjunct corresponds to moving any item `z` that is in the briefcase, thus affecting $At(z,m)$. Ensuring that neither disjunct holds true thus preserves the value of $At(u,v)$ whenever a `MovB` is executed.

McDermott [1991] represented action schema as secondary preconditions to create a provably complete, total-order planner for ADL. While generating these secondary preconditions is straightforward (given the techniques found in Pednault's thesis [1986]) the resulting expressions are often unwieldy, containing multiple disjunctions, equality constraints and functionals. McDermott simplified these formulae through the use of heuristic rules for efficiency purposes. The resulting algorithm, however, sacrificed completeness and produced anomalous behavior for some simple tasks.

We chose a different approach. Our key insight was to first separate the logical connectives and relations from the metric functions and equality constraints. Next, we realised that generating a complete, disjunctive, secondary precondition was unnecessary in many circumstances. For example, not all disjuncts might be required for a particular planning task. We then devised an algorithm that constructed the secondary preconditions dynamically, introducing constraints and logical connectives only when absolutely necessary.

2.3 Steps and Effects

Instead of representing an action schemata as a complex set of secondary preconditions, we convert action schemata into canonical tuples called *steps* and *effects*. These tuples permit UCPOP to quickly identify the relevant parts of each add or delete condition when generating (or extending) a secondary precondition. We believe this results in a considerably more efficient regression process (see the comparison to PEDESTAL in section 6).

Definition *A* STEP S *is a triple* $\prec\rho,\varepsilon,\beta\succ$ *where* ρ, *the step preconditions, is a set of quantified literals;* ε, *the step effects, is a set of effects; and,* β, *the step constraints, is a set of equality constraints on free variables in* ρ.

The notations ρ_S, β_S and ε_S refer to the sets ρ, β and ε within step S. ρ_S are the common preconditions among all effects ε. The equality constraints in β apply to all formulae used in the effects ε as well as the preconditions ρ. The binding constraints β are represented as a set of (possibly negated) pairs, either (u,v) or $\neg(u,v)$. The former indicates that the free variables (or constants) u and v must codesignate, *i.e.*, u and v unify in any well-formed formula. The latter, $\neg(u,v)$ indicate non-codesignation, *i.e.*, u and v can not unify in any well-formed formula. The set of bindings β thus specify an equivalence relation (written "$\approx_\beta$") on plan variables and constants. We will say that a pair (u,v) is CONSISTENT WITH A SET OF BINDINGS β when $u \approx v$ holds true under the relation $\approx_\beta$. Similarly, $\neg(u,v)$ is consistent when $u \approx v$ does not hold true under $\approx_\beta$.

Each effect $e \in \varepsilon_S$ is then represented as follows. The notations ρ_E, β_E and θ_E refer to the sets ρ, β and θ within effect E.

Definition *An* EFFECT E *is a triple* $\prec\rho,\theta,\beta\succ$ *where* ρ, *the effect preconditions, is a set of quantified literals;* θ, *the effect postconditions, is a set of quantified literals; and* β, *the effect binding constraints, is a set of equality constraints on free variables in* ρ *and* θ.

The step and effect tuples clearly distinguish between the logical connectives and equality constraints required by secondary preconditions. When the UCPOP algorithm wants to generate part of a causation precondition for some relation R, it looks for an effect tuple $e = \prec\rho,\theta,\beta\succ$ where $\theta \vdash R$. Recall that the cau-

[5] While our implementation now successfully handles disjunctive preconditions and we have almost finished implementing a class of metric updates, we have not extended our proofs to these cases.

sation precondition Σ_R^A dictates *all* conditions under which an action A will cause R to become true. Since the effect e captures at least one possible way in which the action A can generate R, (*i.e.*, $\theta_e \vdash R$), it must be true that $\rho_e \wedge \beta_e \vdash \Sigma_R^A$. UCPOP can then post the conjuncts of ρ_e as new subgoals and β_e as new constraints that must be met by the plan. If the action A were newly instantiated from an action schema into a step s, UCPOP would also post the subgoals ρ_s and β_s. These new goals represent a nondeterministic choice (*i.e.*, the choice of one disjunct of Σ_R^A), corresponding to the decision to make sure that effect e will be the true cause of R. If this later proves impossible, UCPOP backtracks.

For example, consider a step $\prec\rho, \varepsilon, \beta\succ$ describing the action `MovB(m,l)`. The precondition, $\rho = \{$`At(B,m)`$\}$, specifies that the briefcase must first be at some location `m`. The binding constraints, $\beta = \{\neg($`m,l`$)\}$, require that the origin and destination be distinct. The effects, ε, specify that both the briefcase and its contents move after execution of `MovB(m,l)`. This set ε is written below, following the internal representation of effects as tuples of the form $\prec\rho, \theta, \beta\succ$:

$$\left\{ \begin{array}{l} \forall z \prec\{\texttt{In(z)}\}, \{\neg\texttt{At(z,m)}, \texttt{At(z,l)}\}, \{\neg(\texttt{z,B})\}\succ, \\ \prec\emptyset, \{\neg\texttt{At(B,m)}, \texttt{At(B,l)}\}, \emptyset\succ \end{array} \right\}$$

Note the use of the quantifier $\forall$`z` to indicate that the free variable `z` in the first effect tuple is to be universally, not existentially, quantified.

3 Representing Plans and Problems

With a totally ordered sequence of steps, it is fairly easy to identify the first step that causes some proposition c to be true in the world. We call this step the SOURCE OF c; if c is true in the initial world state, we refer to the dummy step, S_0, as the source. With partially ordered steps, things are more complex and it is useful to explicitly record the intended source for a proposition during planning. Elaborating on the work of Tate [1977], McAllester [1991], and others we define:

Definition *A* CAUSAL LINK *is a quadruple* $\prec S_i, e, r, S_j\succ$, *denoted* $S_i \xrightarrow{e,r} S_j$, *where* r *is a (possibly negated) precondition of* S_j *(or a precondition of one of its effects), and* e *is an effect of* S_i, *and* $\exists q \in \theta_e$ *such that* q *unifies with* r.

To aid in its decision-making, UCPOP maintains a list of causal links from effects of steps to goals and subgoals. These links represent the assumptions upon which a plan P relies and are a crucial aspect of partially ordered PLANS.

Definition *A* PLAN *is quadruple* $\prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ$, *where* $\mathcal{S}$ *is a set of steps,* $\mathcal{B}$ *is a set of binding constraints on free variables in* $\mathcal{S}$, $\mathcal{O}$ *is a set of ordering constraints* $\{S_i < S_j \mid S_i, S_j \in \mathcal{S}\}$, *and* $\mathcal{L}$ *is a set of causal links.*

Pednault's plans [1986] only included steps and ordering constraints. Our representation denotes the importance of separating the equality constraints from all other logical connectives. The equality constraints from secondary preconditions are gathered and kept in a single set, $\mathcal{B}$. No goals are ever posted for satisfying equality constraints. Instead, they are immediately checked against $\mathcal{B}$ for consistency. If at any time during the planning process this set becomes inconsistent, the plan is eliminated from consideration; UCPOP then backtracks. Plan tuples can be used to represent the problem being solved as we now explain.

Definition *A* PLANNING PROBLEM α *is a quadruple* $\prec\Lambda, \mathcal{I}, U, \Gamma\succ$ *where* Λ *is a set of action schemata,* $\mathcal{I}$ *is a set of literals indicating the "initial conditions,"* Γ *is a set of quantified clauses indicating the "goals," and* U *is a universe of discourse for variables in* $\Lambda, \mathcal{I}$, *and* Γ.

To assure systematic establishment of goals and subgoals that have universal quantified clauses, we map these clauses into a set of corresponding ground clauses.

Definition *The* UNIVERSAL BASE Υ *of a first-order clause is defined recursively as follows:*

- $\Upsilon(\Delta) = \Delta$ *if* Δ *contains no quantifiers.*
- $\Upsilon(\exists x_1 \ldots x_n \Delta) = \exists x_1 \ldots x_n \Upsilon(\Delta)$,
- $\Upsilon(\forall x_1 \ldots x_n \exists y_1 \ldots y_n \Delta) = \exists y_1' \ldots y_n' \Upsilon(\Delta')$ *where* Δ' *is a conjunction of formulae identical to* Δ, *one for each possible interpretation of the* x_i *under a universe* U. *The* y_i *are renamed to unique names* y_i'.

For example, suppose that $\Delta = \{\exists x\, Above(x, A), \forall y \exists w\, On(y, w)\}$. Let the universe of discourse for Δ be the pair of blocks, $\{A, B\}$. The universal base $\Upsilon(\Delta)$ is the set of three elements:

$$\{\exists x\, Above(x, A),\ \exists w'\, On(A, w'),\ \exists w'\, On(B, w')\}$$

Each clause within $\Upsilon(\Delta)$ is said to be "universally ground." This reformation of universally quantified clauses is similar to the use of Skolem functions, $y = f(x)$, for existentials y and universals x. Recall that Skolem functions eliminate universals x by replacing existentials y with dummy functions $f(x)$ that produce the corresponding values for each instance of a universal. Whereas Skolem functions are applied from the inside out, the universal base is applied from the outside in.

The universal base of a set of clauses is isomorphic to the image of all Skolem functions over some set of clauses Δ. Instead of using $y = f(x)$, we enumerate the set $\{f(x_0), f(x_1), \ldots, f(x_n)\}$ for each x_i in the range of $f(x)$ and then generate the appropriate set of

clauses Δ' through substitution on the corresponding universal x, then rename y to y'. Renaming prevents name conflicts in alternating quantifiers. We assume that the universe, U, and hence the universal base is finite.

Definition *Let $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ denote a planning problem. The* GOAL PLAN *of α, written* g-plan(α), *is a plan $\prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ$ where $\mathcal{S} = \{S_0, S_\infty\}$, $\mathcal{O} = \{S_0 < S_\infty\}$, $\mathcal{B} = \mathcal{L} = \emptyset$, the initial step, S_0 introduces α's initial conditions but has no preconditions or constraints and the goal step, S_∞, has α's goals as preconditions, but has no effects or binding constraints.*

In the next section, we describe an algorithm, UCPOP, that takes the goal plan of a problem as input and systematically adds steps and constraints until it finds one that solves the planning problem. In section 5 we prove that UCPOP is sound and complete.

4 The UCPOP Planning Algorithm

At the very heart, UCPOP is a theorem prover that resolves step preconditions against effect postconditions. It reduces all quantified formulae into universally ground propositions and then manipulates them to create and extend secondary preconditions. These preconditions are then "satisfied" by creating causal links and then protecting them from effects of other steps. A common subroutine of UCPOP is used throughout the algorithm:

Definition $\texttt{MGU}(p, q)$ *is a function that returns the most general unifier of literals p and q. $\perp$ is returned if no such unifier exists.*

When Δ is a set of clauses, the notation $\Delta\backslash\texttt{MGU}(p, q)$ represents the set of clauses Δ' resulting from applying the substitution $\delta\backslash\texttt{MGU}(p, q)$ to every clause $\delta \in \Delta$. The form of a general unifier is taken to be a set of pairs $\{(u, v)\}$, indicating that u and v must codesignate to ensure that p and q unify. This will allow us to treat binding constraints β as a conjunction of general unifiers.

For example,

$$\begin{aligned} \texttt{MGU}(In(P), In(x)) &= \{(x, P)\} \\ \texttt{MGU}(In(D), In(D)) &= \emptyset \\ \texttt{MGU}(At(x, y), In(x)) &= \perp \end{aligned}$$

4.1 Algorithm overview

The UCPOP algorithm (figure 2) takes three inputs: a plan $\prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ$, an agenda of outstanding goals G, and a set of action schemata Λ. Each entry on the goal agenda is a pair, $\prec c, S\succ$, in which S denotes a plan step and c denotes a precondition of that step [Pednault 1986]. If c is an equality constraint on variables (u, v), we rename the variables to (u_s, v_s) and then add the

Algorithm UCPOP($P = \prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ, G, \Lambda$)

1. **Termination:** If G is empty, report success and return P.
2. **Goal selection:** Choose a goal $\prec c, S\succ \in G$. If a link $S_i \stackrel{e,\neg c}{\rightarrow} S$ exists in $\mathcal{L}$, fail (an impossible plan). Note that c is universally ground.
3. **Operator selection:** Nondeterministically choose any existing (from $\mathcal{S}$) or new (instantiated from Λ) step S_s with effect e and a universally ground clause $p \in \Upsilon(\theta_e)$ where $\texttt{MGU}(c, p) \neq \perp$. If no such choice exists then fail. Otherwise, let $\mathcal{L}' = \mathcal{L} \cup \{S_s \stackrel{e,c}{\rightarrow} S\}$, $\mathcal{B}' = \mathcal{B} \cup \texttt{MGU}(c, p) \cup \beta_e \cup \beta_{S_s}$, $\mathcal{O}' = \mathcal{O} \cup \{S_s < S\}$, $G' = G - \prec c, S\succ$, and let $\mathcal{S}' = \mathcal{S} \cup \{S_s\}$.
4. **Subgoal generation:** If effect e has not already been used to establish a link in $\mathcal{L}$ with bindings $\texttt{MGU}(c, p)$ then let $G' = G$ and for each $\sigma \in \Upsilon(\rho_e \backslash \texttt{MGU}(c, p))$ add $\prec\sigma, S_s\succ$ to G'. If $S_s \notin \mathcal{S}$, for each $\sigma \in \Upsilon(\rho_S \backslash \texttt{MGU}(c, p))$ also add $\prec\sigma, S_s\succ$ to G'.
5. **Causal link protection:** Let l be a causal link $S_i \stackrel{e,q}{\rightarrow} S_j$ in $\mathcal{L}$. Let S_k be any step with an effect e_k and postcondition $p \in \theta_{e_k}$. Step S_k THREATENS link l with clause $v_p \in \Upsilon(p)$ if possibly $S_i < S_k < S_j$ when $\texttt{MGU}(\neg q, v_p) \neq \perp$ is consistent with $\mathcal{B}$. For all such S_k, e_k, l and v_p such that S_k threatens l with v_p, nondeterministically do one of the following (or, if no choice exists, fail):
   (a) **Promotion** If possibly $S_j < S_k$, let $\mathcal{O}' = \mathcal{O} \cup \{S_j < S_k\}$.
   (b) **Demotion** If possibly $S_k < S_i$, let $\mathcal{O}' = \mathcal{O} \cup \{S_k < S_i\}$.
   (c) **Separation** Let $\mathcal{O}' = \mathcal{O} \cup \{S_i < S_k < S_j\}$ then nondeterministically
      i. Choose constraints β' on existentially quantified variables such that $\texttt{MGU}(\neg q, v_p) = \perp$ and let $\mathcal{B}' = \mathcal{B}' \cup \beta'$, or
      ii. Choose a precondition $r \in \Upsilon(\rho_{\mathcal{B}_k} \backslash \texttt{MGU}(\neg q, v_p))$ and let $G' = G' \cup \{\prec\neg r, S_k\succ\}$.
6. **Recursive invocation:** If $\mathcal{B}'$ is inconsistent then fail; else call UCPOP($\prec\mathcal{S}', \mathcal{B}', \mathcal{O}', \mathcal{L}'\succ, G', \Lambda$).

Figure 2: The UCPOP partial order planning algorithm handles actions with universally quantified preconditions and effects as well as conditional effects yet is sound and complete.

constraint to $\mathcal{B}$ (constants remain unchanged). Variable renaming is done when an action schema is first instantiated as a step in the plan (line 3 of UCPOP), following the use of "fresh variables" in [McAllester and Rosenblitt 1991].

When used at the top level to solve a planning problem $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$, the algorithm is called with α's goal plan, its universally ground goals, and its schemata: UCPOP(g-plan(α), $\Upsilon(\Gamma)$, Λ). Note that while the goal agenda G is a set of tuples $\prec c, S\succ$ where c must be achieved by step S, we sometimes use an abbreviation for top-level goals. For example, we often write $\Upsilon(\Gamma)$ rather than the more cumbersome

$$\{\prec c, S_\infty\succ \mid \forall c \in \Upsilon(\Gamma)\}$$

Our algorithm depends on the following assumptions:

- The initial world state is complete.
- The universe of discourse U is fixed and finite.
- All changes to the world state are dictated by actions and explicitly stated.
- Actions are deterministic.
- Actions are consistent, *i.e.*, no action will add both ϕ and $\neg\phi$ to any consistent world state under any condition.
- All relations R are *regressively ascertainable everywhere* [Pednault 1986], *i.e.*, the truth value of every relation R at some step S_i can be determined solely by the actions and the initial state.

The UCPOP algorithm continually refines an incomplete plan P until all goals and subsequent subgoals are satisfied. These refinements include the addition of new steps to $\mathcal{S}$, the constraining of free variables through additional codesignation bindings in $\mathcal{B}$, and the ordering of steps via constraints in $\mathcal{O}$. Lines 3 and 4 dynamically introduce portions of causation preconditions. Line 5 chooses how to create disjuncts of preservation conditions for some relation q. These rely heavily on the structure of effects and steps.

4.2 Efficiency concerns

The UCPOP algorithm is clearly exponential. To improve efficiency, the UCPOP implementation distinguishes between static and dynamic propositions and avoids copying the static world description in each partial plan. The universe of discourse is implemented as a static, typed hierarchy of LISP objects. Universal quantification is handled "lasily," using an iteration abstraction that dynamically expands universal clauses to cover more and more cases (similar to the plan transformation rule 4.11 in [Pednault 1986]). A closed world assumption is adopted to prevent a large ε_{S_0} that would normally have several $\neg p$ effects. The codesignation constraints β are implemented as monotonically converging equivalence classes, similar to the union-find algorithm. Partial orderings on steps are simple lists of pairs $(S_i, S_j) \iff S_i < S_j$, a departure from the current trend of complex temporal databases. UCPOP uses A^* search (with the same evaluation function) on every space of partial plans. Although there are numerous hooks for domain dependent heuristics, we haven't yet systematically explored their use; we believe that this avenue offers the greatest opportunity for performance improvement.

5 Formal Properties

In this section we prove that UCPOP is sound and complete. We adopt the model-theoretic semantics of ADL and refer the reader to [Pednault 1986] for a complete description.

5.1 Formalising Solutions

Soundness implies that every plan produced by UCPOP is a solution to the original problem. Completeness implies that, if there is a solution to some problem α, UCPOP will find it. Both of these concepts rely on the definition of a SOLUTION. We construct this definition by starting from notions of world states, then defining what it means to execute actions, execute plans, and finally converging on the formal concept of solution.

Recall that states s of the world are algebraic structures, *i.e.*, models in logic. Steps, which are instances of ADL action schemata, are modeled by a set of state pairs $\prec s, t\succ$ where the step can be executed in state s to produce state t. A set of types T describes collections of objects in U. Each type $t_i \in T$ is a fixed set of objects $\{o_0, o_1, \ldots, o_n\} \subseteq U$. Types may overlap; for example, `block` might be a type representing all known blocks in U and `physob` might be a set of all physical objects. Types play an important role when determining the scope of universal quantifiers.

We use a primitive state-based model of time where all actions are atomic and cannot overlap. Thus, the temporal history of the world is represented by a linear sequence of states separated by single actions.

Given these concepts, we now define what it means to execute a step. We have slightly changed the definition in [Pednault 1986] to correspond to our notation and assumptions. Deterministic actions insist that every state s exists in only one pair $\prec s, t\succ$ in an action a. Whereas Pednault assumed a set of initial states, we assume one complete initial state.

Definition *Let $\{S_i\}_{i=0}^n$ be a sequence of steps S_i and let $\mathcal{W}$ be a state of the world. The* RESULT OF EXECUTING *$\{S_i\}_{i=0}^n$ IN $\mathcal{W}$, written* RESULT*$(\{S_i\}_{i=0}^n \mathcal{W})$, is defined recursively as follows:*

- RESULT$(\{S_i\}_{i=0}^0, \mathcal{W}) = \mathcal{W}$
- RESULT$(\{S_i\}_{i=0}^n, \mathcal{W}) = t$ *such that*

$\prec$RESULT$(\{S_i\}_{i=0}^{n-1}, \mathcal{W}), t\succ$ is a state pair in the action S_n.

The previous definition says nothing about whether it is feasible to "execute" an action in some state. It only details what happens to a state s when the action is applied to produce state t. We now define the notion of executability for partially ordered plans. First, we define a mapping from partially ordered to totally ordered plans:

Definition *Let $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ denote a planning problem. A totally ordered sequence of steps $\{S_i\}_{i=0}^{n}$ is a* GROUND TOPOLOGICAL SORT *of a plan $\prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ$ if there is a bijection, f, from $\mathcal{S}$ to the steps in $\{S_i\}_{i=0}^{n}$, f is a homomorphism with respect to the ordering constraints $\mathcal{O}$, there exists a global substitution Θ that binds all unbound variables and is consistent with $\mathcal{B}$, and for every step S in the plan, $f(S) = S\backslash\Theta$.*

Given this, we can map partially ordered plans into sets of totally ordered plans, and then define what it means to execute each one:

Definition *A step $A = \prec\rho, \varepsilon, \beta\succ$, representing a set of possible actions, is* EXECUTABLE IN A STATE *s if and only if $s \models \rho_A$ and β_A is consistent with s. A plan $P = \prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ$ is* EXECUTABLE IN INITIAL STATE *$\mathcal{I}$ if, for every ground topological sort $\{S_i\}_{i=0}^{n}$ of P, each step S_i is executable in state* RESULT$(\{S_i\}_{i=0}^{i-1}, \mathcal{I})$.

Recall that a planning problem is a collection of action schemata, initial conditions, universe, and goals.

Definition *A plan $Q = \prec\mathcal{S}_q, \mathcal{B}_q, \mathcal{O}_q, \mathcal{L}_q\succ$ is an* ENHANCEMENT *of a plan $P = \prec\mathcal{S}_p, \mathcal{B}_p, \mathcal{O}_p, \mathcal{L}_p\succ$ for a planning problem $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ if and only if $\mathcal{S}_p \subseteq \mathcal{S}_q$, $\mathcal{O}_p \subseteq \mathcal{O}_q$, $\mathcal{B}_p \subseteq \mathcal{B}_q$, and $\mathcal{L}_p \subseteq \mathcal{L}_q$.*

Definition *A* SOLUTION *to a planning problem $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ is a plan $P = \prec\mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L}\succ$ where P is an enhancement of* g-plan(α), *P is executable in $\mathcal{W}$, and all $S_i \in \mathcal{S}$ are instances of action schemata in Λ.*

Note that if P is a solution, it follows that RESULT$(\{S_i\}_{i=0}^{n}, \mathcal{I}) \models \Gamma$ since $\rho_\infty = \Gamma$ and S_∞ is executable.

5.2 Soundness

Our proof of soundness relies heavily on Pednault's causality theorem [1986] which is akin to a version of Chapman's modal truth criterion [Chapman 1987] for plans with conditional actions. We restate the causality theorem here for convenience:

Theorem 1 Pednault's Causality Theorem. *A condition φ will be true at a point p during the execution of a plan if and only if one of the following holds:*

1. *An action a is executed prior to point p such that*
   (a) *Σ_φ^a is true immediately before executing a.*
   (b) *Π_φ^b is true immediately before the execution of each action b between a and point p.*
2. *φ is true in the initial state and Π_φ^a is true immediately before the execution of each action a prior to point p.*

Proving soundness means demonstrating that the UCPOP algorithm is correct, *i.e.*, that every answer from UCPOP(g-plan(α), $\Upsilon(\Gamma)$, Λ) is a correct solution to the planning problem.

We use a standard technique for recursive algorithms, proving that a loop invariant holds before and after every recursion. We use this to show that this invariant holds whenever UCPOP halts. The invariant we use is defined as follows.

Definition (The UCPOP Loop Invariant) *If the subgoals in the goal agenda G are satisfied by P, then P will be a solution to α.*

Lemma 2 *The* UCPOP *loop invariant holds on the initial call to* UCPOP(g-plan(α), $\Upsilon(\Gamma)$, Λ).

Proof. Trivially, if the goals $G = \Upsilon(\Gamma)$ are satisfied for step S_∞, then Γ is satisfied and P would be a solution. □

Lemma 3 *If the loop invariant holds before an iteration of* UCPOP, *it will hold afterwards.*

Proof. By corollary 3.29 of Pednault's thesis [1986], we can replace goals in G (a subset of his "agenda") with the causation preconditions of steps in P whose effects achieve those goals and the preservation preconditions of steps that might threaten the goals. Then, by Pednault's causality theorem [Pednault 1991], if these preconditions are satisfied, the original goals G are satisfied. We now argue that UCPOP correctly performs these goal transformations.

The condition φ and the "point p" of the causality theorem match the variable c and the step S from line 1 of UCPOP. The action a refers to the new or existing step S_s from line 2 of UCPOP. Note that the new steps S_s are instantiated from Λ and that this is the only place where new steps are introduced. Also, note that case 2 of the causality theorem is handled by choosing step S_0 in line 2 of UCPOP.

UCPOP diverges from a direct, procedural encoding of the causality theorem as a result of the following observation. Instead of requiring that the entire causation preconditions Σ_φ^a be generated and posted as a subgoal, it is sufficient to post another condition ρ that subsumes the more complex formulae, *i.e.*, where $\rho \vdash \Sigma_\varphi^a$. These ρ constraints are exactly those formulae

stored in step and effect tuples which, mentioned earlier, are syntactic transformations of action schemata. Following the assumptions of UCPOP we can translate the causation preconditions from Pednault's thesis and rewrite them as follows:

$$\Sigma^a_\varphi = \bigvee_{\forall e \in \varepsilon_a \mid \theta_e \vdash \varphi} \rho_e \wedge \beta_e$$

$\Sigma^a_{x=y}$ = False $\leftrightarrow$ x and y are unique constants, True otherwise.

$\Sigma^a_{x\neq y}$ = False $\leftrightarrow$ x and y are the same constant, True otherwise.

The operator selection line of UCPOP chooses an effect e where $\texttt{MGU}(c,p) \neq \bot$ and $p \in \Upsilon(\theta_e)$. Since $\texttt{MGU}(c,p)$ returns a general unifier and $\Upsilon(\theta_e)$ returns a set of formulae p such that $\theta_e \vdash p$, if $\texttt{MGU}(c,p) \neq \bot$ then $\theta_e \vdash c$. Thus, substituting φ for c, UCPOP correctly chooses one disjunct, for effect e, of the secondary precondition Σ^a_φ. Since UCPOP works with action schema, it dynamically adjusts the set of actions a represented by the schema by also introducing the constraints $\texttt{MGU}(c,p)$. Following corollary 3.29, UCPOP replaces the original goal c with ρ_e (line 4) and records this decision as a causal link $S_e \xrightarrow{e,c} S$.

Equality constraints $a = b$ for the step S, which are derived by a disjunction of β_e's, are handled as follows. UCPOP renames the variables to $a_s = b_s$ and then adds them to $\mathcal{B}$ (line 3, $\mathcal{B}' = \mathcal{B} \cup \texttt{MGU}(c,p) \cup \beta_e \cup \beta_{S_e}$). If $\mathcal{B}$ ever becomes inconsistent, at least one of the equality "goals" is not met. This mimics the secondary precondition $\Sigma^a_{x=y}$ = False. Otherwise, any grounding of the free variables in a consistent set of constraints $\mathcal{B}$ would satisfy the equality goals, subsuming $\Sigma^a_{x=y}$ = True. Thus $\mathcal{B}$ is a syntactic translation of goals of the form $a = b$ at some step S and UCPOP correctly posts this subset of the causation preconditions for step S_e and effect e. This depends upon our assumption that all free variables are existential and that all constants are unique. We conclude that UCPOP correctly handles case (1), subcase (a) of the causality theorem.

Case (1), subcase (b) and case (2) of the causality theorem require preservation preconditions for all steps b that possibly come between step S_e and S in the plan that requires c to be true at S. Again, we rewrite these preconditions in terms of our assumptions and tuples, as follows:

$$\Pi^a_\varphi = \bigwedge_{\forall e \in \varepsilon_a \mid \theta_e \vdash \varphi} \neg\rho_e \vee \neg\beta_e$$

$\Pi^a_{x=y}$ = False $\leftrightarrow$ x and y are the same constant, True otherwise.

$\Pi^a_{x\neq y}$ = False $\leftrightarrow$ x and y are unique constants, True otherwise.

UCPOP handles subcase b as follows. It either (1) introduces additional step ordering such that subcase b no longer applies, or (2) follows the previous intuition and posts goals $\neg q$ such that $\neg q \vdash \Pi^a_\varphi$ for all such actions a. Lines 5a and 5b of UCPOP handle the former approach. Line 5c, in conjunction with the conditions of line 5, handle the latter (following the previous argument for causation preconditions).

By [Pednault 1986, corollary 3.29], satisfying the newly revised goal agenda G' is equivalent to satisfying G. UCPOP only recurses when $\mathcal{B}'$ is consistent and it only introduces new steps from Λ. Since the loop invariant holds before the execution of UCPOP every ground topological sort of the original plan P would be executable if G were satisfied. By corollary 3.29 and the fact that the action preconditions for S_e are introduced in line 4, P' must also be executable if the new G' were satisfied. Since UCPOP monotonically increases the set of binding constraints, causal links and steps, each iteration returns an enhancement of g-plan(α). Thus, P would be a solution if G' were true. □

Lemma 4 *If the loop invariant holds before an iteration of* UCPOP *and then* UCPOP *halts, the invariant still holds.*

Proof. UCPOP halts in line 1 when the goal agenda G is empty, without performing any modifications. Since no structures are modified the loop invariant must still hold after UCPOP halts.

Theorem 5 (Soundness) *Let* $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ *be a planning problem. If* $\text{UCPOP}(\text{g-plan}(\alpha), \Upsilon(\Gamma), \Lambda)$ *returns a plan* P *then* P *is a solution for* α.

Proof. The previous lemmas combine to form a simple inductive argument that the UCPOP loop invariant holds whenever UCPOP halts provided that it is invoked with $\text{UCPOP}(\text{g-plan}(\alpha), \Upsilon(\Gamma), \Lambda)$. Since the goal agenda is empty when UCPOP halts, no additional conditions are required for P to be a solution. Thus, any plan P returned by $\text{UCPOP}(\text{g-plan}(\alpha), \Upsilon(\Gamma), \Lambda)$ must be a solution to α in and of itself. □

5.3 Completeness

Before stating the completeness theorem, we present a few useful corollaries of the soundness theorem.

Corollary 5.1 *Let* $\alpha = \prec\Lambda, \mathcal{I}, U, \Gamma\succ$ *be a planning problem. If* $\{S_i\}_{i=0}^n$ *is a solution to* α *then for* $2 < k < n$, $\{S_i\}_{i=k}^n$ *is a solution to* $\prec\Lambda$, $\textsc{Result}(\{S_i\}_{i=0}^{k-1}, \text{I}), U, \Gamma\succ$.

In other words: if an n step plan is a solution to a planning problem with initial conditions $\mathcal{I}$, then the last $n-1$ steps are a solution to the modified planning problem which starts with the initial conditions derived by executing the first step in $\mathcal{I}$.

Proof. Follows directly from the definition of RESULT$(\{S_i\}_{i=0}^n, \mathcal{I})$. □

Definition *A plan $P = \prec \mathcal{S}, \mathcal{B}, \mathcal{O}, \mathcal{L} \succ$ is* FULLY SUPPORTED *if every step $S_i \in \mathcal{S}$ is fully supported. A step S is* FULLY SUPPORTED *in P if its preconditions ρ_S are fully supported. A precondition r of a step S_j is* FULLY SUPPORTED *in P if there is a causal link $S_i \xrightarrow{e,r} S_j \in \mathcal{L}$, $S_i < S_j \in \mathcal{O}$ and effect e is fully supported. An effect e of step S is* FULLY SUPPORTED *in P if S is fully supported and every precondition $r \in \rho_e$ is fully supported.*

Corollary 5.2 *Let $\alpha = \prec \Lambda, \mathcal{I}, U, \Gamma \succ$ be a planning problem. If* UCPOP(g-plan(α), $\Upsilon(\Gamma)$, Λ) *returns a plan P then P is fully supported.*

Proof. Follows from the soundness theorem. Note that each link (line 3) records decisions made by UCPOP as to which steps and effects satisfy which preconditions. □

Theorem 6 (Completeness) *Let $\alpha = \prec \Lambda, \mathcal{I}, U, \Gamma \succ$ be a planning problem and let $\{S_i\}_{i=0}^n$ be a solution to α with no extra steps. With the appropriate search strategy* UCPOP *applied to* g-plan(α) *will return a solution, P, such that $\{S_i\}_{i=0}^n$ is a ground topological sort of P.*

Proof. We use induction on the number of steps in the totally ordered solution. To finesse issues of search control, we use $\{S_i\}_{i=0}^n$ as an oracle to guide the construction of the partially ordered plan; McDermott [1991] refers to this as a clairvoyant algorithm. A* or IDA* search [Korf 1985] is used to maintain completeness in our implementation.

Base Case: $\{S_i\}_{i=0}^k \wedge k = 2$.

This means that no plan steps (besides the dummy initial and goal steps) are necessary to achieve the goal, *i.e.*, $\mathcal{I} \vdash \Gamma$. Thus a call to clairvoyant UCPOP(g-plan(α), $\Upsilon(\Gamma)$, Λ) will result in operator selection (step 3) consistently choosing S_0 as the establishing step and making causal links solely to this initial step. Since no new steps are added, none of the links can be threatened. Thus UCPOP will terminate with a solution, P, that has no additional steps, satisfying our need to have the sequence be a ground topological sort of P.

Inductive Step: $\{S_i\}_{i=0}^k \wedge k > 2$.

Given a solution $\{S_i\}_{i=0}^n$ with at most $k-1$ steps, *i.e.*, $n < k$, we assume inductively that clairvoyant UCPOP correctly generates a solution, P, such that $\{S_i\}_{i=0}^n$ is a ground topological sort of an enhancement of P. Now consider the solution $\{S_i\}_{i=0}^k$ to the planning problem $\prec \Lambda, \mathcal{I}, U, \Gamma \succ$; we show that UCPOP finds a corresponding solution here as well.

Let α' be the planning problem $\prec \Lambda$, Result(S_1, $\mathcal{I}$), U, $\Gamma \succ$. By inductive assumption and corollary 5.1, clairvoyant UCPOP(g-plan(α'), $\Upsilon(\Gamma)$, Λ) will return a solution $P_a = \prec \mathcal{S}_a, \mathcal{B}_a, \mathcal{O}_a, \mathcal{L}_a \succ$ when given the $k-1$ step sequence $\{S_i\}_{i=2}^k$ as guidance. P_a is very close to the plan that we are seeking, but the initial step of P_a is doing double duty — it is acting as the source for propositions that either S_0 or S_1 provide in the original totally ordered solution. To distinguish these, we define three subsets of the causal links in $\mathcal{L}_a$.

We first denote a set L_0 of links whose source propositions c remain unchanged from the original initial conditions $\mathcal{I}$ until after the step S_1 is executed. Thus, S_1 does not delete any of these propositions (although it may add duplicates which are absorbed by RESULT$(S_1, \mathcal{I})$):

- $L_0 = \{S_i \xrightarrow{e,c} S_j \mid \quad i = 0 \quad \wedge \quad \mathcal{I} \models c \quad \wedge$ RESULT$(S_1, \mathcal{I}) \models c\}$.

Another, distinct subset of links L_{add} contains links whose source propositions must be added by the execution of step S_1. If a proposition c is not in the initial conditions $\mathcal{I}$, but is in RESULT$(S_1, \mathcal{I})$, and since S_1 is the only step executed between these two situations, c can only be caused by some effect e in S_1:

- Let $L_{add} = \{S_i \xrightarrow{e,c} S_j \mid i = 0 \wedge \mathcal{I} \not\models c \wedge$ RESULT$(S_1, \mathcal{I}) \models c\}$

A third set L_{del} contains links whose source propositions must be "deleted" by the execution of step S_1. If a proposition c was contained in the initial conditions $\mathcal{I}$, but is absent in RESULT$(S_1, \mathcal{I})$, then c must be deleted by one of S_1's effects since S_1 is the only step executed between these two situations:

- Let $L_{del} = \{S_i \xrightarrow{e,c} S_j \mid i = 0 \wedge \mathcal{I} \models c \wedge$ RESULT$(S_1, \mathcal{I}) \not\models c\}$

Now consider the execution trace of all choices made by UCPOP in constructing P_a. We can use this trace, in conjunction with the above sets of links, to carefully guide UCPOP in solving the original problem α, producing a plan with $n+1 = k$ steps. The only difference between α' and α is in the initial state, $\mathcal{I}$. We will thus have to intervene, guiding UCPOP whenever it wants to create a link from the initial state. Otherwise, all goals and threat elimination can proceed as in the execution trace of P_a.

Whenever a link was chosen from L_0 for P_a, UCPOP is guided to create a link from the original, initial conditions captured by an effect in S_0. Whenever a link $S_0 \xrightarrow{e,c} S_j$ was chosen from L_{add} or L_{del} to create P_b, UCPOP is guided to create a corresponding link $S_1 \xrightarrow{e,c} S_j$ from a new step, S_1, which is identical to the step S_1 in the sequence $\{S_i\}_{i=0}^k$. All other links can proceed as before. Finally, after the execution trace from P_a is exhausted, UCPOP will have additional goals of the form

$\prec c, S_1 \succ$. Since S_1 was executable in RESULT$(S_0, \mathcal{I})$, these can be satisfied by links from the initial state $\mathcal{I}$ without posing new threats. UCPOP is then guided to create these links. The final result is an enhancement of g-plan(α) and it must executable in the initial state since it is nearly identical to P_a, except for the step S_1, which we know is executable. Thus, we have constructed a solution to α that contains a ground topological sort $\{S_i\}_{i=0}^{k}$. Thus, UCPOP is complete. □

6 Related Work

Our work was directly motivated by that of Chapman [1987] and McAllester and Rosenblitt [1991] on the foundations of partial order STRIPS planning. The basic SNLP algorithm [Barrett *et al.* 1991], based on the work of McAllester, was derived from Chapman's TWEAK [1987] and Tate's NONLIN [1977]. These partial-order planners use action representations based on the STRIPS formalism. Chien [Chien and DeJong 1992] introduced conditionals into TWEAK and proved an incremental convergence to soundness. The work is parallel to our efforts on proving UCPOP sound.

Pednault provides an elegant theoretical foundation for total order planning with ADL in [Pednault 1986,Pednault 1988]. This work is extended to handle partial order plans in [Pednault 1991]. Although UCPOP was developed independently from [Pednault 1991], pages 243–244 of that document provide an informal description of lines 3–5 of our algorithm. Pednault's theory of planning and action transcend our implementation as it encompasses incomplete initial states, nondeterministic actions, functional updates, and disjunctive preconditions. While Pednault did pose a rule-based, plan enhancement algorithm, no complete implementation of his theory has yet been attempted. We have selected a large subset of this language, but are still far from total coverage.

McDermott's Pedestal [1991] was the first implementation of a planner using ADL, but Pedestal used a total order plan representation. McDermott proved a simple version of his algorithm complete, but frustration with performance issues led him to pursue heuristic variations that sacrificed completeness. We believe that the fundamental problem with Pedestal is its brute force generation of entire preservation and causation preconditions at runtime. Since these formulae tend to be fraught with redundant formulae and constraints, heuristic simplification is the only recourse.

Instead of generating the entire secondary preconditions as suggested by Pednault [Pednault 1986] and implemented by McDermott [1991], UCPOP divides these preconditions into separate logical, equality, (and in our current extensions) metric functional aspects; this allows specialised solvers to deal with the constraints in an optimized fashion.

Our work follows that of Collins and Pryor [1992] where they extended SNLP to handle conditional effects. They did not, however, consider universal quantification in either preconditions or effects nor did they prove soundness or completeness.

7 Future Plans

We have enhanced the algorithm to handle disjunctive preconditions and have almost completed the extension which allows UPDATES of monotonic metric functions. A high priority is to extend our formal results to the augmented algorithm. We also wish to allow actions that create and destroy objects in the universe of discourse U.

We are surprised by UCPOP's performance on the simple problems on which we have tested it and we plan to investigate performance issues more thoroughly. Doubtlessly, UCPOP's speed is largely a function of the simplicity of the classic test problems. Although we have described our algorithm nondeterministically, any actual implementation of it must search. It is manifest that good search techniques are the key to efficient planning. As a result, we believe that UCPOP affords an excellent platform for experimentation with search control heuristics [Barrett 1992,Knoblock 1991] and speed-up learning techniques [Minton 1988,Etzioni 1990b,Etzioni 1990a]. We have therefore developed a model and language for supporting search control heuristics in UCPOP (similar to that in PRODIGY [Minton *et al.* 1989]). Although fully implemented, our model for search control is still being tested; we plan to report our findings in the near future.

8 Conclusions

This paper presents a clean and elegant algorithm for partial order planning with an expressive action representation. UCPOP handles a large subset of ADL, including actions with conditional effects, universally quantified preconditions and effects, and universally quantified goals. We prove that UCPOP is sound and complete and briefly describe our full implementation. We believe that UCPOP's simplicity and efficient implementation makes it an excellent vehicle for further research on planning and learning.

ACKNOWLEDGMENTS

This research was funded in part by National Science Foundation Grant IRI-8957302, Office of Naval Research Grant 90-J-1904, and a grant from the Xerox corporation. Ed Pednault made many detailed and helpful suggestions; we also acknowledge useful discussions with Tony Barrett, Benjamin Grosof, and Steve Hanks.

References

[Barrett and Weld 1992] A. Barrett and D. Weld. Partial Order Planning: Evaluating Possible Efficiency Gains. Technical Report 92-05-01, University of Washington, Department of Computer Science and Engineering, July 1992.

[Barrett *et al.* 1991] A. Barrett, S. Soderland, and D. Weld. The Effect of Step-Order Representations on Planning. Technical Report 91-05-06, University of Washington, Department of Computer Science and Engineering, June 1991.

[Barrett 1992] A. Barrett. Search-Control Heuristics and Abstraction in Least-Commitment Planning. In *Proceedings of the 1992 Workshop on Problem Reformulation and Representation Change*. NASA technical Report FIA-92-06, May 1992.

[Chapman 1987] D. Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32(3):333–377, July 1987.

[Chien and DeJong 1992] S. Chien and G. DeJong. Incremental Reasoning in Explanation-based Learning of Plans: A Method and Evaluation. In *Proceedings of AAAI-92*, August 1992.

[Collins and Pryor 1992] G. Collins and L. Pryor. Achieving the functionality of filter conditions in a partial order planner. In *Proceedings of AAAI-92*, August 1992.

[Etzioni 1990a] Oren Etzioni. *A Structural Theory of Explanation-Based Learning*. PhD thesis, Carnegie Mellon University, 1990. Available as technical report CMU-CS-90-185.

[Etzioni 1990b] Oren Etzioni. Why Prodigy/EBL works. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 1990.

[Fikes and Nilsson 1971] R. Fikes and N. Nilsson. STRIPS: A new Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3/4), 1971.

[Kautz 1982] H. Kautz. A first order dynamic logic for planning. Tech Rept CSRG-144, Department of Computer Science, University of Toronto, 1982.

[Knoblock 1991] C. Knoblock. *Automatically Generating Abstractions for Problem Solving*. PhD thesis, Carnegie Mellon University, 1991. Available as technical report CMU-CS-91-120.

[Korf 1985] R. Korf. Depth-first iterative deepening: An optimal admissible tree search. *Artificial Intelligence*, 27(1), 1985.

[McAllester and Rosenblitt 1991] D. McAllester and D. Rosenblitt. Systematic Nonlinear Planning. In *Proceedings of AAAI-91*, pages 634–639, July 1991.

[McDermott 1991] D. McDermott. Regression planning. *International Journal of Intelligent Systems*, 6:357–416, 1991.

[Minton *et al.* 1989] Steven Minton, Jaime G. Carbonell, Craig A. Knoblock, Daniel R. Kuokka, Oren Etzioni, and Yolanda Gil. Explanation-based learning: A problem-solving perspective. *Artificial Intelligence*, 40:63–118, 1989. Available as technical report CMU-CS-89-103.

[Minton *et al.* 1991] S. Minton, J. Bresina, and M. Drummond. Commitment Strategies in Planning: A Comparative Analysis. In *Proceedings of IJCAI-91*, August 1991.

[Minton 1988] S. Minton. Quantitative Results Concerning the Utility of Explanation-Based Learning. In *Proceedings of AAAI-88*, pages 564–569, August 1988.

[Pednault 1986] E.P.D. Pednault. *Toward a mathematical theory of plan synthesis*. PhD thesis, Stanford University, December 1986.

[Pednault 1988] E.P.D. Pednault. Synthesising plans that contain actions with context-dependent effects. *Computational Intelligence*, 4(4):356–372, 1988.

[Pednault 1989] E.P.D. Pednault. Adl: Exploring the middle ground between strips and the situation calculus. In *Proceedings Knowledge Representation Conf.,*, 1989.

[Pednault 1991] E.P.D. Pednault. Generalising nonlinear planning to handle complex goals and actions with context-dependent effects. In *Proceedings IJCAI-91*, July 1991.

[Rosenschein 1981] S.J. Rosenschein. Plan synthesis: A logical perspective. In *Proceedings of IJCAI-81*, August 1981.

[Sussman 1975] G. Sussman. *A Computer Model of Skill Acquisition*. American Elsevier, New York, 1975.

[Tate 1977] A. Tate. Generating Project Networks. In *Proceedings of IJCAI-77*, pages 888–893, 1977.

[Weld and de Kleer 1989] D. Weld and J. de Kleer, editors. *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufmann, San Mateo, CA, August 1989.

An Approach to Planning with Incomplete Information

Oren Etzioni, Steve Hanks, Daniel Weld,
Denise Draper, Neal Lesh, Mike Williamson*
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195

Abstract

Classical planners presuppose complete and correct information about the world. This paper provides the syntax and semantics for UWL, a representation for goals and actions that facilitates planning with incomplete information about the world's state. While the expressive power of UWL is limited compared to previous work on logics of knowledge and belief, UWL has the advantage of being easily incorporated into planning algorithms. We describe a provably correct planning algorithm based on UWL. To demonstrate UWL's expressive power we encode a subset of the UNIX[1] domain (planning to achieve UNIX goals, using UNIX shell commands as primitive actions), which is difficult to capture using existing planning languages.

1 Introduction and Motivation

Classical planners (*e.g.* [Chapman 1987,Fikes and Nilsson 1971]) presuppose correct and complete information about the world. Assuming correct information means that every proposition entailed by the planner's world model is in fact true in the world. The converse of this assumption is the assumption of complete information: every proposition that is true in the world is entailed by the planner's world model. This paper provides the syntax and semantics of UWL, a representation for goals and actions that does not assume complete information on the part of the planner.[2] A planner might have incomplete information about the world state, its own actions, or exogenous events (*e.g.* the actions of other agents). We focus on incomplete but correct information about the world state.

This departure from classical planning raises a number of fundamental questions:

- How do we represent the goal of obtaining information? (*e.g.* "determine the color of `door-1`") Are "information goals" different from the standard goals used in classical planning?
- How do we represent actions whose primary function is to obtain information rather than to change the world's state (*e.g.* SENSE-COLOR)?
- Actions that change the world provide us with information as well. For example, one way of determining the color of a door is to paint it blue. How do we decide which action to use in order to satisfy an information goal (*e.g.* SENSE-COLOR *versus* PAINT)?
- Can we extend classical planning algorithms to allow for information goals and sensing actions?

*Authors' names are listed alphabetically within two status-oriented equivalence classes. Our research was funded in part by National Science Foundation Grants IRI-8957302 and IRI-9008670, Office of Naval Research Grants 92-J-1946, 90-J-1904, a NASA Graduate Student Researcher's Fellowship, an Office of Naval Research Graduate Fellowship, and a grant from the Xerox corporation. We thank Ernie Davis, Mark Drummond, Keiji Kanazawa, Craig Knoblock, Leora Morgenstern, Alicia Pérez, Rich Segal, and the anonymous reviewers for helpful comments on previous drafts.

[1] UNIX is a trademark of AT&T Bell Labs.

[2] UWL stands for the University of Washington Language.

1.1 A Motivating Example

Consider the problem of satisfying the goal (color door-1 blue). A classical planner will check whether its current state satisfies the goal and, if necessary, create a plan that changes the state to one in which the goal is satisfied. If the planner's world model is incomplete, however, the goal may actually be satisfied in the world but this fact will not be true in the world model. Thus the planner has two options: try to change the world in order to satisfy the goal, or try to elaborate the model by *sensing* whether (color door-1 blue) is true in the world.

Is sensing necessary? Painting the door blue (with no sensing) seems to satisfy the goal, so why bother with sensory operations? There are a number of possible reasons: the planner may have no blue paint, or the blue paint may be needed to achieve a different goal; painting an already-painted door may have a harmful side effect (*e.g.* the door will be covered with noxious wet paint), *etc.* There is a deeper reason, though.

Suppose that the planner is told that the hidden treasure it is seeking is located behind "the blue door." Painting a door blue does not satisfy the goal of finding "the blue door"— it merely obscures the identity of the appropriate door. In general, if a planner is given a definite description that is intended to identify a *particular* object, then changing the world so that another object meets that description is a mistake. The appropriate behavior is to scan the world, leaving the relevant properties of the objects unchanged until the desired object is found.

The goal (color door-1 blue) is ambiguous when provided to a planner with an incomplete world model. Should the planner change the world to make this goal true, or should it sense whether it is true, leaving door-1's color unchanged? UWL provides a way to represent the types of goals and actions involved.

Goals and actions of this sort frequently appear in the UNIX domain (planning to achieve UNIX goals, relying on UNIX commands as primitive actions). An agent's model of its UNIX environment is invariably incomplete, giving rise to information goals such as (active.on neal june.cs.washington.edu) sensory actions (*e.g.*, finger, ls, pwd, *etc.*), and actions that change the world state (rlogin, mv, cd, *etc.*) Thus, the UNIX domain provides a natural test of the expressiveness of UWL.[3]

[3]In fact, the design of UWL was originally motivated by the project of building a *softbot* (software robot) for UNIX [Etzioni and Segal 1992].

1.2 Contributions

The research contribution of the UWL language stems from the following novel features:

- UWL allows for an incomplete world model and for conditional plans.
- UWL actions may change the state of the world, the state of the agent's knowledge, or a combination of the two.
- UWL goals may include specifications of both desired states of knowledge, desired states of the world, and injunctions against changing certain aspects of the world state.
- The language is tractable; we describe an implemented planner for UWL.
- UWL is expressive enough to encode a nontrivial subset of the UNIX domain.

1.3 Organization of the Paper

Section 2 provides the syntax and semantics of UWL. Section 3 describes extensions to the SNLP algorithm [McAllester and Rosenblitt 1991,Barrett and Weld 1992] allowing it to generate plans involving sensory operations. Section 4 demonstrates the expressive power of UWL by encoding a subset of the UNIX domain. The paper concludes with a discussion of related work, the limitations of UWL, and directions for future work.

2 The Language

UWL is an extension of the STRIPS language [Fikes and Nilsson 1971]. Section 2.1 describes the syntax of UWL, Section 2.2 follows with a specification of its semantics. Section 2.3 discusses how to use the language to formalize the notion of an information goal.

2.1 Syntax

A BNF syntax for UWL appears in Table 1. The novel features of the language are: using annotations to distinguish causal effects from observational effects and goals of information from goals of satisfaction, an explicit notion of a proposition's state being neither true nor false, the explicit representation of information that is unknown at plan time but will be provided at run time by sensing actions, and conditional plan steps that exploit this run-time information.

The definitions for planning problems and for a plan that solves a planning problem are typical of STRIPS-like systems: a planning problem consists of an initial state, a goal state, and a set of operator schemas. A plan is a totally ordered set of *steps*, which are ground instances of the operators. Every plan will contain an

planning-problem	::=	**Goal:** *goals* **Initial:** *literal** **Actions:** *operator**
plan	::=	*pcond* | *vcond* | *step plan* | ϵ (the empty plan)
truth-value	::=	**T**| **U**| **F**
pred	::=	a constant symbol designating a predicate name
const	::=	a constant symbol designating something in the world
var	::=	a symbol of the form "?c" designating a variable
rvar	::=	a symbol of the form "!c" designating a run-time variable
vc	::=	*var* | *const*
rvc	::=	*rvar* | *var* | *const*
vtv	::=	*var* | *truth-value*
rvtv	::=	*rvar* | *var* | *truth-value*
content	::=	(*pred vc**)
rcontent	::=	(*pred rvc**)
literal	::=	(*content* . *vtv*)
rliteral	::=	(*rcontent* . *rvtv*)
goals	::=	((**satisfy** *literal*) | (**hands-off** *content*) | (**find-out** *literal*))*
postconditions	::=	((**cause** *literal*) | (**observe** *rliteral*))*
operator	::=	**Name:** *name* **Preconds:** *goals* **Postconds:** *postconditions*
step	::=	an instance of an *operator* with no unbound variables
pcond	::=	(**pcond** *content* (*plan*) (*plan*))
vcond	::=	(**vcond** (*rvar* = *const*) (*plan*) (*plan*))

Table 1: BNF specification of UWL.

initial step and a goal step. The initial step's precondition set is empty and its postcondition set asserts the problem's initial state. The goal step's precondition set consists of the problem's goal state, and it has no postconditions. Defining plans in this way means that we don't need to talk about a problem's goals being satisfied; instead we talk about every step's preconditions being met. This is what we mean by a *correct* plan, and defining correctness is the focus of our discussion of a plan's semantics.

Before moving to the discussion of a plan's semantics we should give intuitive definitions for the syntactic constructs that extend STRIPS: the distinction between causal and observational postconditions, the concept of a run-time variable, the additional truth value for propositions, and the conditional plan constructs.

The first extension involves dividing an operator's effects into those that change the world (the **cause** annotation), and those that change the planner's state of information (the **observe** annotation)[4]. Causal postconditions correspond to STRIPS' adds and deletes.

Observational postconditions come in two forms, corresponding to the two ways the planner can gather information about the world at run time: it can observe the truth value of a proposition, (**observe** ((**P c**) . **!v**)), or it can identify an object that has a particular property, (**observe** ((**P !x**) . **T**)).

The variables **!v** and **!x** are *run-time* variables, used to refer to a piece of information that will not be available until execution time. We require that each **observe** postcondition contain *exactly one* run-time variable and that each **cause** postcondition contain *no* run-time variables.

Conditional plan steps use this run-time information. We introduce two conditional constructs corresponding to the two forms for **observe** postconditions: if the planner observes the truth value of a proposition **P** using a run-time variable **!v**, it can then build a plan conditional on this information using the construct (**pcond P** $\mathcal{P}_1$ $\mathcal{P}_2$) where the plan $\mathcal{P}_1$ will be executed if the proposition turns out to be true at execution time, and $\mathcal{P}_2$ will be executed otherwise.

The second form for **observe** postconditions binds its run-time variable to an object that satisfies a particular property. The planner can condition on *which* object the variable will be bound to using the construct (**vcond** ($!x = K$) $\mathcal{P}_1$ $\mathcal{P}_2$) where $\mathcal{P}_1$ will be executed if the execution system binds variable **!x** to constant **K**; otherwise $\mathcal{P}_2$ will be executed.

Our next extension to STRIPS involves annotating preconditions (and thus goals) with **satisfy**, **hands-off**, or **find-out**. We will discuss these annotations in more detail, but the intuition is that a **satisfy** pre-

[4] Causal postconditions imply a corresponding observation: we assume that if an operator causes **P** to become true the planner *knows* that **P** becomes true.

condition can be achieved by any means, causal or observational. The precondition (`find-out (P . T)`) means roughly that the planner wants to determine that P is true, but does not want to change P's state in doing so. (We may want to discourage the planner from finding out that a chair is blue by painting it blue, for example.)

A precondition of the form (`hands-off P`) is a different sort of constraint: it says nothing about P's truth value, but demands that the plan do nothing to change P's state. Section 2.3 discusses how these annotations relate to the intuitive notion of an information or knowledge goal.

Our final extension to the language extends the truth values a proposition can take on: propositions can be either true T, false F, or "unknown" U. Truth values of U apply to propositions about which the planner has incomplete information: those that are not mentioned in the initial state and which no subsequent plan step has changed.

2.2 Semantics

Our discussion of the meaning of UWL plans centers around a definition of what it means for a plan to be *correct*—informally, that it will actually achieve the goals that it was constructed to achieve. Intuitively a *correct* plan is one in which every precondition is *true*. In a totally ordered ground plan without conditionals or run-time variables, this definition is straightforward:

Definition 1 (Correctness (STRIPS version)) *A plan $\mathcal{P}$ is correct just in case every precondition of every one of its steps is true. A step $\mathcal{S}_j$'s precondition* P *is true just in case there is some step $\mathcal{S}_i$, $i < j$ that has* (P . T) *as a postcondition, and there is no intervening step $\mathcal{S}_k$, $i < k < j$ that has* (P . F) *as a postcondition.*

We need to extend this definition to include conditional plans, explicit truth values, and precondition and postcondition annotations.

2.2.1 Plan branches

The STRIPS definition of correctness admits only one course of execution for the plan: $\mathcal{S}_1$, $\mathcal{S}_2$, ... Introducing conditionals means that the actual course of execution may not be known at plan time. We therefore introduce the idea of a plan's *branches*. A branch through a plan is a sequence of non-conditional steps, representing one possible course of execution. Associated with a plan $\mathcal{P}$ is a set of branches, branches($\mathcal{P}$), representing *all possible* courses of execution. We define a plan's branch set using the four possible definitions of a plan from Table 1:

1. branches(ϵ) = $\emptyset$
2. branches($\mathcal{S}$ $\mathcal{P}$) = $\{\mathcal{S}; b \mid b \in \text{branches}(\mathcal{P})\}$
3. branches((pcond P $\mathcal{P}_1$ $\mathcal{P}_2$)) = $\{\mathcal{S}_P; b \mid b \in \text{branches}(\mathcal{P}_1)\} \cup \{\mathcal{S}_{\overline{P}}; b \mid b \in \text{branches}(\mathcal{P}_2)\}$ where $\mathcal{S}_P$ is a step with no preconditions and the single postcondition (`observe (P . T)`) and where $\mathcal{S}_{\overline{P}}$ is a step with no preconditions and the single postcondition (`observe (P . F)`).
4. branches((vcond (!x = K) $\mathcal{P}_1$ $\mathcal{P}_2$)) = $\{\mathcal{S}; b \mid b \in \text{branches}(\mathcal{P}_1)\} \cup \text{branches}(\mathcal{P}_2)$ where $\mathcal{S}$ is a step with no preconditions and the single postcondition (`observe ((!x = K) . T)`).[5]

Now each branch of the plan is a totally ordered ground sequence of steps; we will say that a plan $\mathcal{P}$ is correct just in case every branch in branches($\mathcal{P}$) is correct.

2.2.2 Matching and truth values

We still cannot apply the STRIPS definition of correctness to a plan's branches for three reasons:

1. Preconditions and postconditions have explicit truth values: an operator can require that P be *false* or cause its truth value to change to *unknown*.
2. A precondition can be satisfied through the binding of a run-time variable: a step $\mathcal{S}_k$'s precondition (P K) may be satisfied by a prior step $\mathcal{S}_i$ with postcondition (P !x) and an intervening step $\mathcal{S}_j$ with postcondition (!x = K).
3. Preconditions can have annotations that restrict the form of the steps that can appear in the plan. `find-out` preconditions will generally be satisfied by `observe` postconditions, for example.

We address the second complication by defining what it means for two propositions (presumably one step's precondition and another step's postcondition) to *match*:

Definition 2 (Matching) *Suppose that $\mathcal{S}_i$ has a postcondition whose propositional content is* P_1, *and $\mathcal{S}_k$ has a precondition whose propositional content is* P_2, *and $i < k$.* P_1 matches P_2 *if*

- P_1 *and* P_2 *are identical: the predicates are the same and in every argument place they have exactly the same constant or run-time variable, or*
- P_1 *and* P_2 *are identical except that* P_1 *has a run-time variable* !x *in an argument place, and* P_2 *has a constant* K *in the corresponding argument place, and there is a step $\mathcal{S}_j$, $i < j < k$ with a*

[5] Steps with postconditions of this form occur *only* as a result of "unfolding" conditionals.

postcondition of the form ((!x = K) . T), *and there is no step occurring between j and k with a postcondition of the form* ((!x = L) . T) *for any constant* L.

We now deal with the problem of defining what it means for a plan containing explicit truth values and annotations to be correct. Recall that a precondition will be annotated with one of **satisfy**, **hands-off**, or **find-out**, and a postcondition will be annotated with **cause** or **observe**. Our precondition annotations are intended to convey the following information:

- (**satisfy** (P . v)) Make P have truth value v by any means—causal, observational, or some combination.
- (**hands-off** P) Do not change the value of proposition P.
- (**find-out** (P . v)) Ascertain whether or not P's truth value is v. This can be accomplished in one of two ways:
  - By using a step that has a *observational* postcondition that matches P, or
  - By using a step that has a *causal* postcondition that matches P, as long as that step serves some other purpose in the plan.

The tricky part is to formalize the meaning of "serves some other purpose." To do so we start with the idea of a postcondition *supporting* a precondition. The first part of the definition is similar to the STRIPS notion of a postcondition making a precondition true.

Definition 3 (Postcond. supports precond.) *Suppose that step i has a postcondition with annotation a_i, propositional content p_i and truth value t_i and that step k has a precondition with annotation a_k, propositional content p_k, and truth value t_k. Step i's postcondition* supports *step k's precondition only if*

1. $i < k$,
2. p_i *matches* p_k *(in the sense defined above),*
3. $t_i = t_k$, *and*
4. *there is not step* j—$i < j < k$—*that has a postcondition whose propositional content matches* p_k.

There is one way in which a precondition can be supported without a supporting step *i*: a truth value of U can be satisfied if there is *no* step that affects the proposition.

Definition 4 (Precondition supported) *A step k's precondition is* supported *just in case either*

1. *There is a step i with a postcondition that supports the precondition (as defined above), or*
2. *The precondition's truth value t_k is* U *and there is* no *step $i < k$ with a postcondition that matches the precondition's proposition p_k.*

The definitions so far do not mention the precondition's annotations, so we will define a "valid" precondition to be one that is supported in a manner that respects its annotation. We define what acceptable support is for each of the three precondition annotations.

Definition 5 (Valid precondition) *Consider a precondition of step k with annotation a_k, proposition p_k, and truth value t_k*[6]

1. *If* a_k = **satisfy** *then the precondition is valid just in case it is supported.*
2. *If* a_k = **hands-off** *then the precondition is valid just in case there is no step $i < k$ with a postcondition with annotation* **cause** *that matches p_k.*
3. *If* a_k = **find-out** *then the precondition is valid just in case either*
   (a) *it is supported by a postcondition whose annotation is* **observe**, *or*
   (b) *it is supported by a postcondition of some step i whose annotation is* **cause**, *but step i also has a postcondition that supports some precondition by satisfying either item 1 or item 3a.*

The idea behind the definition for **find-out** preconditions is to disallow a step appearing in a plan if its only purpose is to support **find-out** preconditions using its causal postconditions. If the step is in the plan for some other reason—either because it causes some other proposition that needs to be satisfied or because it observes some proposition that needs to be satisfied or found out—it can then validate the **find-out** precondition as well.

The definition of a correct plan follows directly from the definition of a plan's branches and the definition of a valid precondition:

Definition 6 (Correct plan) *A plan's branch is correct just in case every precondition of every step in the branch is valid. A plan is correct just in case every one of its branches is correct.*

2.3 Discussion

One of the most fundamental questions we address is what is the precise meaning of the informal notion of an "information goal." UWL provides two alternatives amenable to precise specification and simple implementation, but there are certainly many more.

[6] If a_k = **hands-off** then t_k is undefined.

Our first alternative is to represent the information goal "determine that the proposition P is true" as the conjunction (**satisfy** (P . T)) and (**hands-off** P). This encoding can be too restrictive, in some cases, in that it disallows plans in which P is fortuitously achieved. Suppose that a plan contains a step S_i that was inserted into the plan because it achieved goal G, and that step also caused P to be true as a side effect. Defining information goals in terms of **hands-off** means that this plan does *not* satisfy the goal (**satisfy** (G . T)) and (**satisfy** (P . T)) and (**hands-off** P), even though at the end of the plan it is the case that G is true and that P's truth value is known to be T. The problem is that S_i *causes* P to be true, even though the step served to satisfy the other goal as well, and that violates the **hands-off** annotation.

To illustrate this sort of difficulty, consider the goal of printing the file **paper.tex**. The final step in a plan to achieve this goal is **lpr**, the UNIX command that sends a postscript file to the printer. One of the preconditions to **lpr** determines that the argument to the command is indeed a file. If we express this precondition as the conjunction (**satisfy** (**isa file.object ?file**) . T) and (**hands-off** (**isa file.object ?file**)), then we effectively disallow the creation of the postscript file at an earlier step. Yet, that is precisely what we need to do (via the commands **latex** and **dvi-ps**). A more appropriate encoding of the precondition is (**find-out** (**isa file.object ?file**) . T).

In general, we can represent an information goal simply as (**find-out** (P . v)). This encoding is less restrictive since, as in the above example, P's truth value can change as a side effect of achieving a different goal. Neither alternative is guaranteed to be satisfactory in all cases, and UWL is not committed to using either one exclusively. In the UNIX domain we have found that **find-out** is useful for writing operator preconditions and that the **hands-off** and **satisfy** combination is useful for expressing top-level goals (see Table 3). Developing a complete taxonomy of information goals is an area of future research.

3 Partial-Order Planning

We have worked to keep UWL close to the familiar STRIPS representation in order to build on the long-standing body of work on planning algorithms using that representation (see [Allen *et al.* 1990] for a survey). In this section we present SENS$_P$ a provably sound partial-order planning algorithm for UWL based on SNLP [McAllester and Rosenblitt 1991,Barrett and Weld 1992,Hanks and Weld 1992]. We believe SENS$_P$ is complete subject to one simplifying assumption.

The basic operation of SENS$_P$ follows that of SNLP, but a number of important extensions are made to handle the features of UWL. In this paper, we focus on two issues: extending the notion of a *causal link* to planning with incomplete information, and generating conditional plans.

3.1 Causal Links

SENS$_P$ inherits the notion of a causal link from SNLP and earlier planners (*e.g.* [Warren 1974,Tate 1977]). Causal links are used to record why a step was introduced into a plan and to prevent other steps from interfering with that purpose. If a step S_i achieves a proposition p to satisfy a precondition of step S_j, that dependency is recorded by the causal link $S_i \xrightarrow{p} S_j$.

Following McAllester, we say that a link $S_i \xrightarrow{p} S_j$ is *threatened* if some step S_k might be ordered between S_i and S_j, and S_k has a **cause** postcondition that matches p according to the plan's variable-binding constraints. A precondition p of a plan-step S_j is an *open condition* of the plan if there is no causal link $S_i \xrightarrow{p} S_j$. A plan is said to be *complete* if it has no open conditions and no threatened links. SNLP is sound, systematic, and complete (as long as backtracking explores all nondeterministic choice points using a strategy such as iterative deepening) [McAllester and Rosenblitt 1991].

The SENS$_P$ algorithm extends the notion of a casual link to handle the UWL's annotated preconditions and postconditions. The postconditions of a UWL operator are split into a **cause** list and an **observe** list. Since an **observe** postcondition does not change that proposition's state in the world, these conditions cannot pose a threat to any causal links; thus threat detection considers only **cause** postconditions, which correspond to STRIPS add and delete lists.

3.2 Hands-off Goals

SENS$_P$ handles goals of the form (**hands-off** (P **?x**)) by adding a causal link between the plan's initial step and its goal step. The link's "content" is (P **?x**), but the variable **?x** is interpreted as universally quantified. As a result, any step that might be added to the plan that has a **cause** postcondition (P **?y**) (for any variable **?y** or any constant) will be considered a threat to that link, and will not be added to the plan. An **observe** postcondition of (P **?y**), on the other hand, is not considered a threat to the link, and can therefore be added to the plan. Thus **hands-off** preconditions are implemented trivially by exploiting standard techniques for detecting and resolving threats on links.

3.3 Find-out Goals

Find-out preconditions can be achieved by a step's **observe** postcondition or, when the step has some other purpose, by a **cause** postcondition. The actual algorithm to achieve this is moderately complex, but

the two basic ideas are:

1. SENSp branches and separately considers the two ways of establishing the `find-out` goal, backtracking to ensure completeness.
2. To achieve `find-out` goals using a `cause` postcondition the algorithm first tries to add causal links to support all preconditions without `find-out` annotations (adding new steps to the plan if necessary). Then it tries to add links supporting all remaining goals without adding new steps to the plan.

3.4 Run-time Variables

For the purpose of matching preconditions and postconditions, SENSp treats run-time variables as constants whose values are not yet known: they are not allowed to match with other constants or with different run-time variables (*cf.* [Olawsky and Gini 1990]). Run-time variables *can* match ordinary variables, but SENSp adds ordering constraints which ensure that the value of a run-time variable is not used until it has been `observed`.

For example, suppose we wish our kitchen table and chair to share the same color. Further, suppose that while we don't know the color of the table, we don't want to change it. We can express this goal as follows:

```
Goal:
  (satisfy ((color chair ?c) . T))
  (satisfy ((color table ?c) . T))
  (handsoff (color table ?tc))
```

Let there be only three possible actions: we can obtain paint of any color, we can paint any object with any color that we have, and we can sense the color of any object.

```
Name:  (SENSE-COLOR ?obj !color)
Preconds:
Postconds: (observe ((color ?obj !color) . T))

Name:  (GET-PAINT ?color)
Preconds:
Postconds: (cause ((have-color ?color) . T))

Name:  (PAINT ?obj ?color)
Preconds: (satisfy ((have-color ?color) . T))
Postconds: (cause ((color ?obj ?color) . T))
```

SENSp will immediately construct a causal link $S_0 \overset{C_{\texttt{table}}}{\rightarrow} S_\infty$ to enforce the `hands-off` goal.[7] Next, SENSp explores the two ways to satisfy the goal that the table have color `?c`, i.e. by introducing either a PAINT or SENSE-COLOR operation. If SENSp decides to paint the table, it will quickly realize the threat to the `hands-off` link and backtrack. Thus a `(sense-color table !color)` step will be added (which we shall denote as step S_s) along with the causal link $S_s \overset{C_{\texttt{table}}}{\rightarrow} S_\infty$. Since S_s has only `observe` postconditions, it does not threaten the earlier link.

Next, SENSp might try to achieve the fact that the chair has color `?c`. Since the variable `?c` is also mentioned in the goal proposition, `(color table ?c)`, which has been achieved, SENSp has recorded that `?c` must codesignate with the run-time variable `!color` which is bound by the SENSE-COLOR step. In other words, SENSp needs to make the chair have the color corresponding to the value of `!color`. As before, there are two possibilities, PAINT and SENSE-COLOR. If SENSp attempts to use SENSE-COLOR, it will realize that it must construct a conditional plan—we cover this case in the next section, so for now, assume that SENSp chooses to paint the chair. Thus, SENSp adds a PAINT step (S_p) and the causal link $S_p \overset{C_{\texttt{chair}}}{\rightarrow} S_\infty$. Since this PAINT step uses the run-time variable, SENSp constrains $S_s < S_p$.

At this point all the original goals have been supported, but the `paint` step has an unsatisfied precondition, so SENSp needs to achieve `(have-color !color)`. The only way to do this is with a GET-PAINT step, so GET-PAINT is added to the plan as step S_g along with the causal link $S_g \overset{HC}{\rightarrow} S_p$. Since this GET-PAINT step also uses the run-time variable, SENSp constrains $S_s < S_g$. The final plan is thus:[8]

```
(sense-color table !color)
(get-paint !color)
(paint chair !color)
```

3.5 Generating Conditional Branches

SENSp uses a variant of Warren's WARPLAN-C technique for generating conditional plans [Warren 1976]. The basic idea is that conditionals are inserted into the plan only when SENSp needs to constrain the value of a run-time variable. The algorithm must be careful as to how it makes the constraint, however, since it has to obey the requirement that a run-time variable be `observed` before it is used. The algorithm first chooses one value `k` for the run-time variable `!v`, generates a conditional step `(vcond (!v = k) ...)`, then continues planning. It later generates a plan for the other branch (without the equality constraint), then combines the two.[9]

[7] Clarity demands abbreviations in this example; *e.g.*, $C_{\texttt{table}}$ denotes the proposition `(color table ...)`, *etc.*

[8] For brevity, the steps in a plan are indicated by their names only.

[9] Since writing this paper, we have learned of independent work on the synthesis of conditional plans which could

For example, consider a small modification to example above: instead of a completely general GET-PAINT operator, suppose that getting green paint requires a distinct operator, MAKE-GREEN-PAINT, which mixes blue paint and yellow paint to create green. The GET-PAINT operator suffices to get all other colors.[10]

```
Name:  (MAKE-GREEN-PAINT)
Preconds:
  (satisfy ((have-color blue) . T))
  (satisfy ((have-color yellow) . T))
Postconds: (cause ((have-color green) . T))

Name:  (GET-PAINT ?c)
Preconds:
Postconds: (cause ((have-color ?c) . T))
Equals: (<> ?c green)
```

As before, $SENS_P$ will construct the **hands-off** link, and add a sensing operation to determine the color of the table, and as before, let us assume that it adds a PAINT step to paint the table. Now, however, when it attempts to support the precondition (**have-color !color**) for the paint, it will attempt to constrain the run-time variable !color with the constant green. Thus, it will create two copies of the plan, one in which !color is constrained to equal green, and one in which it is constrained not to equal green. The two plans will be independently completed in a manner analogous to the first example, and the merged plan is created by extracting the sensing operation and adding the conditional step (Table 2).

It is tricky to ensure soundness *and* completeness for an algorithm that generates conditional plans. So far we are confident of completeness subject to an important simplifying assumption: that the operator schemata for sensing actions (those with **observe** postconditions) have no preconditions.[11] At this point producing correct plans requires us to have operators, like GET-PAINT, which will work for all cases not covered by more specific operators. In future work, we will consider ways of weakening this restriction, either by producing 'conditionally correct' plans, or by declaring ranges on the possible values of run-time variables.

4 UWL and the UNIX Domain

In [Etzioni and Segal 1992] we describe the project of building *softbots* (*soft*ware ro*bots*): programs that interact with software environments by issuing commands and interpreting the environments' response. We believe that software environments such as operating systems or databases are a pragmatically convenient yet intellectually challenging substrate for AI research. To support this claim, we have developed a softbot for the UNIX operating system that uses UWL to represent its actions (UNIX commands such as **finger** or cd) and goals. The softbot generates and executes plans to achieve the UWL goals it receives as input. The design and development of UWL validates our claim that softbots provide fertile testbeds for AI research. In addition, the ability to encode UNIX commands and goals in UWL demonstrates the language's expressive power (see Table 3 for an illustration). We have represented over twenty UNIX commands as UWL operators, and are in the process of encoding many more.

Models of some UNIX commands (*e.g.*, **cd**, **rm**) can actually be encoded as plain STRIPS operators. However, due to the dynamic nature and sheer size of the UNIX environment, information about it is necessarily incomplete. For example, users continually log in and out, and the number of files accessible through the Internet is staggering. Consequently, many of the most routine UNIX commands (*e.g.*, **ls**, **pwd**, **finger**, **lpq**, **grep**) are used to gather information. Such commands cannot be represented by STRIPS operators but are naturally encoded in UWL (see Table 3).

Information goals arise frequently in the UNIX domain. Consider, for example, the goal of removing a file named **core-dump**. There are several ways to satisfy this goal. One is to find a file named **core-dump**, using **ls**, and remove it. Another is to rename an arbitrary file to **core-dump**, using **mv**, and remove that file. The first alternative is obviously the one intended. UWL forces the correct interpretation by representing the goal as:

(and (**find-out** ((file.name !file core-dump) . **T**))
(**satisfy** ((file.object !file) . **F**)).

The **observe** postcondition of **ls** satisfies the **find-out** goal, whereas the **cause** postcondition of **mv** does not. This goal will only be satisfied when some file named **core-dump** exists. Naturally, stipulating that *all* **core-dump** files should be removed would require universal quantification.

While UWL has turned out to be a convenient language for our softbot, it does not address a number of issues that arise in representing UNIX commands including the need for universally quantified preconditions and postconditions, and the fact that other agents (particularly humans) are continually changing the world's state by logging in and out, creating new files, *etc.* We are currently extending UWL to include universal quantification. We plan to address the dynamic nature of the UNIX domain by allowing the softbot to detect

improve our algorithm; see the discussion of [Peot and Smith 1992] in section 5.

[10] The 'Equals' slot in the last operator description is an extension to UWL that allows codesignation and noncodesignation constraints on variables to be asserted.

[11] This assumption is powerful because it frees $SENS_P$ from subgoaling to achieve observation steps. Subgoaling complicates completeness because the sensing actions can interfere with the steps to be taken after the conditional.

Plan P_T

```
(get-paint yellow)
(get-paint blue)
(sense-color table !color)
(make-green-paint)
(paint chair green)
```

Plan P_F

```
(sense-color table !color)
(get-paint ?c)
(paint chair ?c)
```

Final Plan

```
(sense-color table !color)
(vcond (!color = green)
  (  (get-paint yellow)
     (get-paint blue)
     (make-green-paint)
     (paint chair green))
  (  (get-paint !color)
     (paint chair !color)))
```

Table 2: Subplans for both branches and the final, conditional plan.

and update its incorrect beliefs about the world. In addition, we are developing learning algorithms that will enable the softbot to model the variability of its world over time. Based on this learned model, some of the softbot's observations will persist (*e.g.*, a new workstation has been added to the network) whereas others (*e.g.*, the printer is out of paper) will be forgotten quickly. See [Etzioni *et al.* 1992] for a more comprehensive discussion.

5 Related Work

[McCarthy and Hayes 1969] argues for a formalization of the notion of "knowledge" as the basis for a theory of plans. This approach has led to the development of a rich body of logical work within AI [Morgenstern 1988,Moore 1985].[12] [Moore 1985] introduces a first-order modal logic of knowledge and action, developing the idea of *informative* actions that supply an agent with additional information about the world. [Morgenstern 1987] develops a more expressive theory of action and planning, and specifically addresses the problem of *knowledge preconditions* for the performance of actions and plans. She considers the expression of complex plans using sequential, conditional, iterative, and concurrent constructs, and axiomatizes their knowledge preconditions. She does not, however, address the problem of how these plans might be generated.

[Drummond 1986] presents a framework (plan nets) that allows for the representation of sensory actions. The occurrence of an *event* entails a set of *beliefs*, which is partitioned into two subsets. The *external-results* describe the external, physical effects of the event, while the *internal-results* describe those changes that affect only the agent's world model. However, the precondition and goal languages presented do not provide any way of distinguishing between internal and external results, so the distinction between sensory and non-sensory actions is of limited use. Again, the problem of plan generation is not considered.

[Drummond 1989] presents another version of the plan-net formalism. This version does not make the distinction between sensory and non-sensory effects, but it does provide a rich language for expressing goals. A goal may be any arbitrary boolean combination of propositions, and may also include the meta-predicates *maint(p)* (*p* must be true throughout the plan), *ach(p)* (*p* must become true at some point during the plan), and *obt(p)* (*p* must become true at some point and remain true until the end of the plan). The language is used only for the specification of goals. Although the top-level goals can be annotated, operator preconditions are restricted to be conjunctions of simple propositions.

Our work is close to that of [Olawsky and Gini 1990], which specifically considers the problem of planning with an incomplete initial world-state description. Their approach uses deferred planning: when a proposition's truth value is needed but unknown they suspend the planning process, execute a plan to learn its value, then resume planning. Their action representation is essentially like STRIPS. Although they discuss sensory actions, there is no distinction in their representation between sensory and non-sensory actions. The examples they give are greatly simplified by the fact that their domain contains a single operator that achieves every proposition, which allows them to avoid the complex issues that arise when some proposition is a sensory effect of one action and a causal effect of another.

[Peot and Smith 1992] presents a variant of SNLP for the construction of conditional non-linear plans. Their action representation also uses a three-valued logic, and allows actions to have multiple, mutually-exclusive sets of outcomes. However, they do not provide for a distinction between observational and causal effects of actions, or treat sensing explicitly. It is possible that their planning algorithm could be extended to work with UWL.

Universal plans [Schoppers 1987] and Gapps [Kaelbling 1988] provide methods for constructing exhaustive conditional plans. However, the representation languages employed do not allow explicit description of sensing actions. Rather, it is assumed that complete sensory information is available at every point during the execution of the plan in order to select the correct

[12] Logics of knowledge also appear in the philosophical literature (*e.g.*, [Hintikka 1962]).

UNIX goals:

Determine if neal is on june.cs.washington.edu:

```
Goal:       (find-out ((active.on neal june.cs.washington.edu) .  T))
```

Determine if the file with name "paper.tex" contains the word "theorem:"

```
Goal:       (satisfy ((name ?somefile paper.tex) .  T))
            (hands-off (name ?somefile paper.tex))
            (satisfy ((file.contains.string ?somefile theorem) .  T))
            (hands-off (file.contains.string ?somefile theorem))
```

UNIX operators:

```
Name:       FINGER
Preconds:   (find-out ((isa machine ?machine) .  T))
            (find-out ((isa person ?person) .  T))
Postconds:  (observe ((active.on ?person ?machine) !boolean))

Name:       MV
Preconds:   (find-out ((isa file.object ?file) .  T))
            (find-out ((isa directory.object ?dir1) .  T))
            (find-out ((isa directory.object ?dir2) .  T))
            (find-out ((name ?file ?name) .  T))
            (find-out ((parent.directory ?file ?dir1) .  T))
            (satisfy ((protection ?dir1 readable)) .  T)
            (satisfy ((protection ?dir2 writable)).  T)
            (satisfy ((current.directory softbot ?file ?dir1) .  T))
Postconds:  (cause ((parent.directory ?file ?dir1) .  F))
            (cause ((parent.directory ?file ?dir2) .  T))

Name:       GREP
Preconds:   (find-out ((isa file.object ?file) .  T))
            (find-out ((isa directory.object ?dir) .  T))
            (find-out ((name ?file ?name) .  T))
            (find-out ((parent.directory ?file ?dir) .  T))
            (satisfy ((protection ?file readable) .  T))
            (satisfy ((current.directory softbot ?dir) .  T))
Postconds:  (observe ((file.contains.string ?file ?string) !boolean))

Name:       WC
Preconds:   (find-out ((isa file.object ?file) .  T))
            (find-out ((isa directory.object ?dir) .  T))
            (find-out ((name ?file ?name) .  T))
            (find-out ((parent.directory ?file ?dir) .  T))
            (satisfy ((protection ?file readable) .  T))
            (satisfy ((current.directory softbot ?dir) .  T)) ))
Postconds:  (observe ((character.count ?file !char) .  T))
            (observe ((word.count ?file !word) .  T))
            (observe ((line.count ?file !line) .  T))
```

Table 3: Sample representations of UNIX goals and operators.

action.

[Ram and Hunter 1992] discusses the application of *knowledge goals* as a means of controlling inference. They develop a theoretical framework for describing the explicit desire for knowledge, and illustrate it on examples from natural language understanding and machine learning.

6 Conclusion

This paper contains two fundamental observations. First, we showed that information goals and information-gathering actions can be represented in a simple and elegant manner by annotating the preconditions and postconditions to standard STRIPS operators. The proximity of UWL to the STRIPS language enabled us to extend the SNLP planning algorithm to one that generates correct plans in the presence of incomplete information. In addition, we have developed a softbot that relies on UWL to represent a subset of the UNIX domain (see Table 3 for an illustration). The softbot generates and executes plans to achieve a wide range of UNIX goals, demonstrating the utility and expressiveness of UWL [Etzioni *et al.* 1992]. Second, we showed that posing an information goal to a planner implicitly dictates that the state of the proposition in question should be protected in the process of planning to achieve that goal. This observation is critical to choosing actions appropriately in domains (such as the UNIX domain) that contain both information-gathering actions and actions that change the world's state. We discussed two ways to encode information goals in UWL; future work will explore other ways to do so.

References

[Allen *et al.* 1990] J. Allen, J. Hendler, and A. Tate, editors. *Readings in Planning.* Morgan Kaufmann, San Mateo, CA, August 1990.

[Barrett and Weld 1992] A. Barrett and D. Weld. Partial Order Planning: Evaluating Possible Efficiency Gains. Technical Report 92-05-01, University of Washington, Department of Computer Science and Engineering, July 1992.

[Chapman 1987] D. Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32(3):333–377, July 1987.

[Drummond 1986] M. Drummond. A representation of action and belief for automatic planning systems. In *Proceedings of the 1986 workshop on Reasoning about Actions and Plans* , San Mateo, CA, 1986. Morgan Kaufmann.

[Drummond 1989] M. Drummond. Situated Control Rules. In *Proceedings of the First International Conference on Knowledge Representation and Reasoning*, May 1989.

[Etzioni and Segal 1992] Oren Etzioni and Richard Segal. Softbots as testbeds for machine learning. In *Working Notes of the AAAI Spring Symposium on Knowledge Assimilation*, Menlo Park, CA, 1992. AAAI Press.

[Etzioni *et al.* 1992] Oren Etzioni, Neal Lesh, and Richard Segal. Building softbots for UNIX. In preparation, 1992.

[Fikes and Nilsson 1971] R. Fikes and N. Nilsson. STRIPS: A new Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3/4), 1971.

[Hanks and Weld 1992] Steven Hanks and Daniel Weld. Systematic adaptation for case-based planning. In *Proceedings of the First International Conference on AI Planning Systems*, June 1992.

[Hintikka 1962] Jaako Hintikka. *Semantics for Propositional Attitudes.* Cornell University Press, 1962.

[Kaelbling 1988] Leslie Pack Kaelbling. Goals as parallel program specifications. In *Proceedings of the Seventh National Conference on Artificial Intelligence.* Morgan Kaufmann, 1988.

[McAllester and Rosenblitt 1991] D. McAllester and D. Rosenblitt. Systematic Nonlinear Planning. In *Proceedings of AAAI-91*, pages 634–639, July 1991.

[McCarthy and Hayes 1969] J. McCarthy and P. J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[Moore 1985] R.C. Moore. A Formal Theory of Knowledge and Action. In *Formal Theories of the Commonsense World.* Ablex, 1985.

[Morgenstern 1987] Leora Morgenstern. Knowledge preconditions for actions and plans. In *Proceedings of IJCAI-87*, 1987.

[Morgenstern 1988] Leora Morgenstern. *Foundations of a Logic of Knowledge, Action, and Communication.* PhD thesis, New York University, 1988.

[Olawsky and Gini 1990] D. Olawsky and M. Gini. Deferred planning and sensor use. In *Proceedings, DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control.* Morgan Kaufmann, 1990.

[Peot and Smith 1992] Mark A. Peot and David E. Smith. Conditional nonlinear planning. In *Proceedings of the First International Conference on AI Planning Systems*, June 1992.

[Ram and Hunter 1992] Ashwin Ram and Lawrence Hunter. The use of explicit goals for knowledge to guide inference and learning. Git-cc-92/04, Georgia Institute of Technology, 1992.

[Schoppers 1987] M. Schoppers. Universal Plans for Reactive Robots in Unpredictable Environments. In *Proceedings of IJCAI-87*, pages 1039–1046, August 1987.

[Tate 1977] A. Tate. Generating Project Networks. In *Proceedings of IJCAI-77*, pages 888–893, 1977.

[Warren 1974] D. Warren. WARPLAN: A system for generating plans. Memo No. 76, Univerity of Edinburgh, Department of Computational Logic, 1974.

[Warren 1976] D. Warren. Generating copnditional plans and programs. In *Proceedings of AISB Summer Conference*, pages 344–354, University of Edinburgh, 1976.

Equivalence and Tractability Results for SAS$^+$ Planning

Christer Bäckström
Dept. of Computer and Information Science,
Linköping University
S-581 83 Linköping,
Sweden
email: cba@ida.liu.se

Abstract

We define the SAS$^+$ planning formalism, which generalizes the previously presented SAS formalism. The SAS$^+$ formalism is compared with some better-known propositional-planning formalisms with respect to expressiveness. Contrary to intuition, all formalisms turn out to be equally expressive in a very strong sense. We further present the SAS$^+$-PUS planning problem which generalizes the previously presented, tractable SAS-PUS problem. We prove that also the SAS$^+$-PUS problem is tractable by devising a provably correct polynomial time algorithm for this problem.

1 Introduction

Much effort has gone into finding more and more general formalisms, mainly logic-based, for plans and actions and also into finding reasoning methods for these. Although such formalisms may be important for modelling problems and comparing different approaches we most probably have to identify subproblems and devise tailored algorithms for these in order to overcome the computational difficulties involved. This approach has also been advocated by Bylander [1991b] and Sandewall [1992].

Results on the computational complexity of planning has been practically non-existent until some five years ago. Chapman [1987] proved that first-order planning is semidecidable. Dean and Boddy [1988] proved NP-hardness of temporal projection, even under severe restrictions, and claimed these results to carry over also to plan validation and planning. However, see Nebel and Bäckström [1992] and Bäckström and Nebel [1992] for a discussion of these results. Bylander [1991a] investigated the complexity of propositional STRIPS planning under certain restrictions and found that plan existence is PSPACE-complete in the general case. He reported a few tractable subcases, however. Erol *et al.* [1992a, 1992b] extended the results by Chapman and Bylander and presented a more systematic and complete tabulation of complexity results for STRIPS plan existence under various restrictions. Korf [1987] analysed how the search space can be pruned when the problem enjoys certain properties, but there is never an exponential speed-up in any of these cases. Furthermore, hierarchical abstraction and task reduction can reduce the complexity significantly under certain conditions [Korf, 1987, Knoblock, 1990], but see Bacchus and Yang [1992]. Finally, there are many results on the complexity of temporal reasoning (see, for example, Golumbic [1992]).

Somewhat in contrast to most planning research within AI we have focussed on planning problems where the action representation is relatively simple, but where the problem size makes computational complexity an important issue. Such problems occur, for example, in *sequential control.*[1] Although our results are generally applicable sequential control is our main intended application area. There seems to be mutual benefits from combining research in AI planning and sequential control. On the one hand many problems in sequential control seem to be planning problems. On the other hand application problems in sequential control can gain new insight into which formal restrictions on planning problems are relevant to real applications. In fact, the restrictions used in the SAS$^+$-PUS problem (Section 4) stem from a toy example within sequential control. When applying planning to sequential control problems we will have to emphasize provable correctness and tractability more than has usually been the done in AI [Passino and Antsaklis, 1989, Åström *et al.*, 1991].

We have concentrated on propositional planning since many, if not even most, planning problems can be adequately modelled using finite domains. By diverting

[1] Sequential control is a subdiscipline of the area *discrete event dynamic systems* (DEDS) within *automatic control.* So far there are no 'standard' methods for solving DEDS problems. A brief discussion of some methods that have been attempted appears in Klein and Bäckström [1991].

slightly from the traditional STRIPS formalism and by drawing inspiration from sequential control problems we have managed to identify some problem restrictions that are different in nature from those used by other researchers. We have previously advocated a bottom-up research strategy, starting with some tractable problem and generalizing this as much as possible by removing or replacing restrictions, while retaining tractability. We started with the SAS-PUBS problem [Bäckström and Klein, 1991b] which we generalized to the SAS-PUS problem [Bäckström and Klein, 1991a]. The SAS$^+$-PUS problem presented in this paper is a further step along this strategy. We have been using a non-standard formalism, the SAS$^+$ formalism, which we have now managed to compare formally to some more 'standard' formalisms.

The remainder of this paper is organized as follows. First we present the SAS$^+$ formalism. Then we show that this formalism is equally expressive as some other 'standard' formalisms for propositional planning. Finally, we show that the previously presented, tractable SAS-PUS problem can be generalized to the new, more expressive formalism, yielding the SAS$^+$-PUS problem, which is also tractable. The formal definitions and proofs must be kept brief in this paper, but all details and full proofs appear in Bäckström [1992].

2 The SAS$^+$ Formalism

2.1 World Modelling

We use the **simplified action structures (SAS$^+$)** formalism, which is a slight modification of the traditional STRIPS formalism [Fikes and Nilsson, 1971]. There are mainly two details that differ between the SAS$^+$ formalism and the STRIPS formalism. Instead of propositional atoms we use *multi-valued state variables* and instead of using only pre- and post-conditions for actions we also use a *prevail-condition*. The SAS$^+$ formalism is actually a restricted version of the more expressive *action structures* formalism [Sandewall and Rönnquist, 1986, Bäckström, 1988a, 1988b].

A multi-valued, discrete state variable can, in principle, be replaced by a number of two-valued state variables (propositional atoms). However, there are at least three good reasons for using multi-valued state variables. Firstly, many applications are more naturally described in this way, especially in *sequential control*. Secondly, we have been able to identify certain restrictions on planning problems that seem relevant to real problems and that reduce the computational complexity considerably, but that would not have been easy to identify using a standard formalism. Thirdly, it will be a smaller step to generalize the state variables to structured domains like the integers or interval valued domains [Bäckström, 1988a, 1988b]. To the contrary, a propositional atom is easily modelled as a two-valued state variable.

The prevail-condition of an action expresses what the action requires to hold during its whole occurrence, but which it does not affect. For sequential plans there is, in principle, no need to distinguish the prevail-condition from the pre-condition, but it has conceptual advantages to do so. Although not an issue in this paper, we are also concerned with parallel plans, where the prevail-conditions play an important role [Bäckström and Klein, 1991a, 1991b].

Definition 2.1 A **SAS base-structure** $\Phi = \langle \mathcal{M}, \mathcal{S}, \mathcal{H} \rangle$ is defined by:

- a set $\mathcal{M} = \{i_1, \ldots, i_m\}$ of **state variable indices**;
- a **space** $\mathcal{S} = \mathcal{S}_{i_1} \times \ldots \times \mathcal{S}_{i_m}$ **of total states**, where for each $j \in \mathcal{M}$,
  - $\mathcal{S}_j$ is a **domain** of mutually exclusive values for the jth state variable and
  - $\mathcal{S}_j^+ = \mathcal{S}_j \cup \{\mathsf{u}\}$ is the **extended domain** for the jth state variable,where u denotes the **undefined value**,

  and the **space** $\mathcal{S}^+ = \mathcal{S}_{i_1}^+ \times \ldots \times \mathcal{S}_{i_m}^+$ **of partial states** is implicitly defined;
- a set of **action types** $\mathcal{H}$ where each $h \in \mathcal{H}$ is of the form $h = \langle \mathsf{b}, \mathsf{e}, \mathsf{f} \rangle \in \mathcal{S}^+ \times \mathcal{S}^+ \times \mathcal{S}^+$ such that
  - b denotes the **pre-condition** of h,
  - e denotes the **post-condition** of h and
  - f denotes the **prevail-condition** of h. □

We write $s[i]$ to denote the value of the ith state variable in a state s. We also write $s \sqsubseteq t$ if the state s is subsumed (or satisfied) by state t, *ie.*,

$$s \sqsubseteq t \text{ iff } \forall i \in \mathcal{M}, s[i] = \mathsf{u} \text{ or } s[i] = t[i].$$

An **action** is an *instance* (or occurrence) of an action type. There may be several distinct actions of the same type.

We define a *SAS$^+$-structure* as a SAS base-structure with certain additional restrictions. Firstly, we require that all state variables that have a defined value in the pre-condition of an action type must have a defined but different value in the post-condition (R1). That is, an action cannot change a state variable from a defined value to the undefined value and it must either change a variable or not define it at all in the pre- and post-conditions. Variables that are defined but not changed should go into the prevail-condition. Secondly, the post-condition and the prevail-condition of an action type must not define the same state variables since an action cannot both change a state variable and require it to be constant (R2).

Definition 2.2 A SAS base-structure $\Phi = \langle \mathcal{M}, \mathcal{S}, \mathcal{H} \rangle$ is a **SAS$^+$-structure** iff the set of action types $\mathcal{H}$ satisfies the following restrictions:

(R1) for all $h \in \mathcal{H}$ and for all $i \in \mathcal{M}$, if $\mathsf{b}(h)[i] \neq \mathsf{u}$, then $\mathsf{b}(h)[i] \neq \mathsf{e}(h)[i] \neq \mathsf{u}$

(R2) for all $h \in \mathcal{H}$ and $i \in \mathcal{M}$, either $\mathsf{e}(h)[i] = \mathsf{u}$ or $\mathsf{f}(h)[i] = \mathsf{u}$. □

2.2 Example: The Blocks World

In this section we model the blocks world in the SAS$^+$ formalism in order to give some intuition for the formalism. To keep the example small we assume there are only three blocks: A, B and C. For each block we introduce two state variables: one telling the position of the block and one telling whether the block is clear, that is, whether there is some other block on top of it. The 'clear' variable is used to avoid having to quantify over the positions of all other blocks. We define the set of state variable indices as

$$\mathcal{M}_{BW} = \{PosA, ClrB, PosB, ClrB, PosC, ClrC\}.$$

The corresponding state variable domains are $\mathcal{S}_{PosA} = \{\mathsf{B}, \mathsf{C}, \mathsf{T}\}$, $\mathcal{S}_{PosB} = \{\mathsf{A}, \mathsf{C}, \mathsf{T}\}$ and $\mathcal{S}_{PosC} = \{\mathsf{A}, \mathsf{B}, \mathsf{T}\}$, where A, B, C and T denote block A, block B, block C and the table respectively, and $\mathcal{S}_{ClrA} = \mathcal{S}_{ClrB} = \mathcal{S}_{ClrC} = \{\mathsf{F}, \mathsf{T}\}$, where F and T denote false and true respectively. We let $\mathcal{S}_{BW}$ denote the corresponding set of total states. We write states as tuples of the form

$$\langle PosA, ClrB, PosB, ClrB, PosC, ClrC \rangle$$

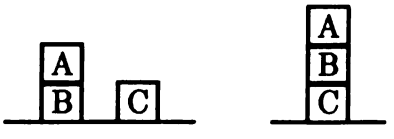
where the indices mark the positions of the corresponding variable values. For instance, the scenarios in Figure 1 are represented as the states

$$s_0 = \langle \mathsf{B}, \mathsf{T}, \mathsf{T}, \mathsf{F}, \mathsf{T}, \mathsf{T} \rangle$$

and

$$s_\star = \langle \mathsf{B}, \mathsf{T}, \mathsf{C}, \mathsf{F}, \mathsf{T}, \mathsf{F} \rangle$$

respectively.



(a) The initial state s_0 (b) The goal state $s_\star$

Figure 1: Some sample blocks world scenarios

Table 1 describes the action types for moving block A. AfromBtoC moves A from the top of B to the top of C. The position of A is changed from B to C, the clear status of B is changed to true and the clear status of C must be true before moving A and will be false after the action. We also require that A is clear during the whole action since we cannot move a block that has some other block on top of itself. The other action types for moving block A are defined analogously.

To make the example somewhat more interesting we introduce some additional action types. We allow two blocks to be moved simultaneously if one is on top of the other. The action types *DmvABTC* moves the blocks A and B, as a two-block stack, from the table to the top of block C. We define other action types moving two blocks in the same way. Some sample such action types are shown in Table 2. Note that ABfromTtoC is not the same action type as BAfromTtoC. We also introduce the action Shake, which shakes the whole table, with the effect that all blocks end up lying directly on the table.

2.3 Plans

Definition 2.3 A **non-linear plan** over a $\Phi = \langle \mathcal{M}, \mathcal{S}, \mathcal{H} \rangle$ is a pair $\langle \mathcal{A}, \prec \rangle$ such that $\mathcal{A}$ is a set of actions over $\mathcal{H}$ and $\prec$ is a strict partial order on $\mathcal{A}$. □

Given a plan $\Delta = \langle \mathcal{A}, \prec \rangle$, we write $CS(\Delta)$ to denote the set of all action sequences $\bar{\alpha} = \langle a_1, \ldots, a_n \rangle$ such that $\{a_1, \ldots, a_n\} = \mathcal{A}$ and $\bar{\alpha}$ is a linearization of Δ, *ie.*, if $a_k \prec a_l$, then $k < l$. Given an action sequence $\bar{\alpha} = \langle a_1, \ldots, a_n \rangle$ and an action a, we define $(\bar{\alpha};a) = \langle a_1, \ldots, a_n, a \rangle$.

Definition 2.4 Given two states $s, t \in \mathcal{S}^+$, we define for all $i \in \mathcal{M}$,

$$(s \oplus t)[i] = \begin{cases} t[i] & \text{if } t[i] \neq \mathsf{u} \\ s[i] & \text{otherwise.} \end{cases}$$

The function *result* gives the state resulting from executing an action sequence and is defined recursively as:

$$\mathit{result}(s, \langle\rangle) = s$$

$$\mathit{result}(s, (\bar{\alpha};a)) = \begin{cases} \mathit{result}(s, \bar{\alpha}) \oplus \mathsf{e}(a) & \text{if } \mathsf{b}(a) \sqsubseteq \mathit{result}(s, \bar{\alpha}) \\ & \text{and } \mathsf{f}(a) \sqsubseteq \mathit{result}(s, \bar{\alpha}) \\ \langle \mathsf{u}, \ldots, \mathsf{u} \rangle & \text{otherwise.} \end{cases}$$ □

This definition of the *result* function 'solves' the *frame problem* [Hayes, 1981, Brown, 1987] by employing the *STRIPS assumption*, which is sufficient in this restricted formalism.

An action a is **admissible** wrt. a state s iff $\mathsf{b}(a) \sqsubseteq s$ and $\mathsf{f}(a) \sqsubseteq s$. An action sequence $\bar{\alpha} = \langle a_1, \ldots, a_n \rangle$ is admissible wrt. a state s iff a_1 is admissible wrt. s and for $1 < k \leq n$, a_k is admissible wrt. $\mathit{result}(s, \langle a_1, \ldots, a_{k-1} \rangle)$. In addition, the empty action sequence is admissible wrt. any state. We also define

$$ACS(\Delta, s) = \{\bar{\alpha} \in CS(\Delta) \mid \bar{\alpha} \text{ is admissible wrt. } s\}.$$

action type	b	e	f
AfromBtoC	⟨B, u, u, F, u, T⟩	⟨C, u, u, T, u, F⟩	⟨u, T, u, u, u, u⟩
AfromCtoB	⟨C, u, u, T, u, F⟩	⟨B, u, u, F, u, T⟩	⟨u, T, u, u, u, u⟩
AfromBtoT	⟨B, u, u, F, u, u⟩	⟨T, u, u, T, u, u⟩	⟨u, T, u, u, u, u⟩
AfromCtoT	⟨C, u, u, u, u, F⟩	⟨T, u, u, u, u, T⟩	⟨u, T, u, u, u, u⟩
AfromTtoB	⟨T, u, u, T, u, u⟩	⟨B, u, u, F, u, u⟩	⟨u, T, u, u, u, u⟩
AfromTtoC	⟨T, u, u, u, u, T⟩	⟨C, u, u, u, u, F⟩	⟨u, T, u, u, u, u⟩

Table 1: Action types for moving block A in the blocks world example.

action type	b	e	f
ABfromTtoC	⟨u, u, T, u, u, T⟩	⟨u, u, C, u, u, F⟩	⟨B, T, u, u, u, u⟩
BAfromTtoC	⟨T, u, u, u, u, T⟩	⟨C, u, u, u, u, F⟩	⟨u, u, A, T, u, u⟩
Shake	⟨u, u, u, u, u, u⟩	⟨T, T, T, T, T, T⟩	⟨u, u, u, u, u, u⟩

Table 2: Some more complex action types for the blocks world.

Definition 2.5 A **SAS$^+$ planning problem instance** is a tuple $\Pi = \langle \Phi, s_0, s_\star \rangle$ such that Φ is a SAS$^+$-structure and $s_0, s_\star \in \mathcal{S}^+$ denote the **initial state** and **goal state** respectively. A plan Δ over Φ **solves** Π iff

1. $ACS(\Delta, s) = CS(\Delta)$ and
2. $s_\star \sqsubseteq result(s_0, \bar{\alpha})$ for all $\bar{\alpha} \in CS(\Delta)$.

More specifically we distinguish four different problems. The **SAS$^+$plan existence problem** is: given an instance Π, decide whether there exists some plan Δ over Φ such that Δ solves Π. The **SAS$^+$plan search problem** is: given an instance Π, find a plan Δ over Φ that solves Π, or answer that there is no such plan. We also define the **bounded** plan existence (search) problem as the plan existence (search) problem with the additional requirement that only plans $\langle \mathcal{A}, \prec \rangle$ such that $|\mathcal{A}| \leq K$ for some parameter K given together with Π count as solutions. □

Furthermore, we say that a plan Δ is a **minimal plan** solving an instance Π if there is no other plan solving Π that contains fewer actions than Δ.

3 Three STRIPS Formalisms for Propositional Planning

In this section we will compare the SAS$^+$ formalism with three better-known formalisms that all derive more directly from STRIPS [Fikes and Nilsson, 1971]. We call these formalisms *Classic Propositional STRIPS* (CPS), *Propositional STRIPS with Negative goals* (PSN) and *Ground TWEAK* (GT).

The CPS formalism is essentially STRIPS restricted such that both the initial and goal states as well as the pre-condition and the add- and delete-lists are sets of propositional atoms and no further formulae may be derived within a state. The PSN formalism generalizes the CPS formalism by also allowing negative pre-conditions and negative goals. This formalism appears, for example, in Bylander [1991a]. The GT formalism is the TWEAK formalism [Chapman, 1987] restricted to ground literals. This formalism generalizes the PSN formalism by also allowing partial initial states. We define these formalisms as follows.

Definition 3.1 An instance of the **PSN planning problem** is given by a tuple $\Pi = \langle \langle \mathcal{P}, \mathcal{O} \rangle, \mathcal{I}, \mathcal{G} \rangle$ where

- $\mathcal{P}$ is a finite set of **atoms**;
- $\mathcal{O}$ is a finite set of **operators** of the form $\langle \varphi, \eta, \alpha, \delta \rangle$ where
  - $\varphi \subseteq \mathcal{P}$ is the **positive precondition**,
  - $\eta \subseteq \mathcal{P}$ is the **negative precondition**,
  - $\alpha \subseteq \mathcal{P}$ is the **positive postcondition** (the *add-list*),
  - $\delta \subseteq \mathcal{P}$ is the **negative postcondition** (the *delete-list*),
  - $\varphi \cap \eta = \alpha \cap \delta = \emptyset$;
- $\mathcal{I} \subseteq \mathcal{P}$ is the **initial state**;
- $\mathcal{G} = \langle \mathcal{G}^+, \mathcal{G}^- \rangle$ is the **goal** where
  - $\mathcal{G}^+ \subseteq \mathcal{P}$ is the **positive goal**,
  - $\mathcal{G}^- \subseteq \mathcal{P}$ is the **negative goal**,
  - $\mathcal{G}^+ \cap \mathcal{G}^- = \emptyset$. □

An instantiation of a PSN operator is called a **PSN action**. Given a PSN action a we write $\varphi(a)$ to denote the pre-condition of the corresponding operator

and analogously for the other three conditions. The result of executing a sequence of PSN actions is defined recursively as

$$result_{PSN}(S, \langle\rangle) = S,$$
$$result_{PSN}(S, (\bar{\alpha};a)) = \begin{cases} T \cup \alpha(a) - \delta(a) & \text{if } \varphi(a) \subseteq T \text{ and} \\ & \eta(a) \cap T \neq \emptyset, \\ T & \text{otherwise,} \end{cases}$$

where T is an abbreviation for $result_{PSN}(S, \bar{\alpha})$.

A PSN action a is **PSN admissible** wrt. a state $S \subseteq \mathcal{P}$ iff $\varphi(a) \subseteq S$ and $\eta(a) \cap S = \emptyset$. PSN admissible action sequences and $ACS_{PSN}(S, \Delta)$ is defined correspondingly.

Definition 3.2 Given an instance $\Pi = \langle\langle \mathcal{P}, \mathcal{O}\rangle, \mathcal{I}, \mathcal{G}\rangle$ of the PSN planning problem, a plan $\Delta = \langle \mathcal{A}, \prec\rangle$ such that $\mathcal{A}$ is a set of PSN actions **solves** Π iff both

1. $ACS_{PSN}(\mathcal{I}, \Delta) = CS(\Delta)$;
2. for all $\bar{\alpha} \in CS(\Delta)$, both
   (a) $\mathcal{G}^+ \subseteq result_{PSN}(\mathcal{I}, \bar{\alpha})$,
   (b) $\mathcal{G}^- \cap result_{PSN}(\mathcal{I}, \bar{\alpha}) = \emptyset$. □

Instances of the **CPS problem** can be viewed as instances of the PSN problem having $\mathcal{G}^- = \emptyset$ and $\eta = \emptyset$ for all operators.

Definition 3.3 An instance of the **GT planning problem** is given by a tuple $\Pi = \langle\langle \mathcal{P}, \mathcal{O}\rangle, \mathcal{I}, \mathcal{G}\rangle$ where

- $\mathcal{P}$ is a finite set of **atoms**, implicitly defining a finite set of **literals** $\mathcal{L} = \mathcal{P} \cup \{\neg p \mid p \in \mathcal{P}\}$;
- $\mathcal{O}$ is a finite set of **operators** of the form $\langle pre, post\rangle$ where
  - $pre \subseteq \mathcal{L}$ is the **precondition**,
  - $post \subseteq \mathcal{L}$ is the **postcondition**,
  - pre and $post$ are consistent and $pre \cap post = \emptyset$;
- $\mathcal{I} \subseteq \mathcal{L}$ is the **initial state**;
- $\mathcal{G} \subseteq \mathcal{L}$ is the **goal**;
- $\mathcal{I}$ and $\mathcal{G}$ are consistent,

where a state $S \subseteq \mathcal{L}$ is **consistent** iff there is no $p \in \mathcal{P}$ such that both $p \in S$ and $\neg p \in S$. □

The concept **GT action** and the abbreviations $pre(a)$ and $post(a)$ for GT actions are defined analogously to the PSN case. The function $Neg : 2^{\mathcal{L}} \to 2^{\mathcal{L}}$ is defined for states such that

$$Neg(S) = \{p \mid \neg p \in S\} \cup \{\neg p \mid p \in S\},$$

	CPS	PSN	GT	SAS$^+$
Partial goals	•	•	•	•
Partial init. states			•	•
Neg. preconds.		•	•	•
Neg. goals		•	•	•
Multi-valued state variables				•

Table 3: A comparison of the CPS, PSN, GT and SAS$^+$ formalisms.

that is, $Neg(S)$ denotes the *negation* of S. The result of executing a sequence of GT actions is defined recursively as

$$result_{GT}(S, \langle\rangle) = S,$$
$$result_{GT}(S, (\bar{\alpha};a)) = \begin{cases} T \cup post(a) & \\ \quad - Neg(post(a)) & \text{if } pre(a) \subseteq T, \\ T & \text{otherwise,} \end{cases}$$

where T is an abbreviation for $result_{GT}(S, \bar{\alpha})$.

A GT action is **GT admissible** wrt. a consistent state $S \subseteq \mathcal{L}$ iff $pre(a) \subseteq S$. GT admissible action sequences and the notation $ACS_{GT}(S, \Delta)$ are defined analogously to the PSN case.

Definition 3.4 Given an instance $\Pi = \langle\langle \mathcal{P}, \mathcal{O}\rangle, \mathcal{I}, \mathcal{G}\rangle$ of the GT planning problem, a plan $\Delta = \langle \mathcal{A}, \prec\rangle$ such that $\mathcal{A}$ is a set of GT actions **solves** Π iff both

1. $ACS_{GT}(\mathcal{I}, \Delta) = CS(\Delta)$,
2. $\mathcal{G} \subseteq result_{GT}(\mathcal{I}, \bar{\alpha})$ for all $\bar{\alpha} \in CS(\Delta)$. □

The SAS$^+$ formalism introduced in Section 2 can be viewed as the GT formalism generalized to multi-valued state variables instead of literals.

A comparison of the four formalisms appears in Table 3. It seems that the CPS, PSN, GT and SAS$^+$ formalisms are successively more and more expressive, in this order. This turns out not to be the case, however. All four formalisms are in fact equally expressive for sequential planning. Below we sketch the proof that the plan existence problems in these formalism polynomially reduce to each other, thus establishing an equivalence result. This does not seem quite sufficient for claiming that the formalisms are equally expressive, however, since this allows for some anomalous cases. Suppose there are two search problems $\mathcal{X}$ and $\mathcal{Y}$ (for instance, the plan search problem expressed in two different formalisms) such that their corresponding decision (existence) problems polynomially reduce to each other. Although the problems reduce to each other it may be the case that the instances of $\mathcal{Y}$ have minimal solutions that are exponentially bigger than the solutions for the corresponding instances of $\mathcal{X}$. That is,

although the decision problems polynomially reduce to each other, the search problems do not. To handle such anomalous situations without losing generality we do not consider two problems equally expressive unless also the solutions for corresponding instances are of size polynomially bounded in each other. The full proof [Bäckström, 1992, Chapter 5] uses a novel type of reduction (three-stage polynomial reduction) to cope with this problem.

Theorem 3.5 The plan existence problem expressed in either of the CPS, PSN, GT or SAS$^+$ formalisms can be polynomially reduced to the plan existence problem expressed in any of the other three formalisms. □

Proof sketch: We prove the existence of the polynomial reductions a–d in Figure 3.[2] The reductions a, b and c are straightforward. The only non-trivial reduction is d, which can be done in two steps, first reducing SAS$^+$ plan existence to GT plan existence and then reduce the latter to CPS plan existence. Any state variable can be simulated by a number of propositional atoms. However, in order to assure that the reduction is polynomial we may use only $O(\log n)$ atoms for a domain of size n. This can be achieved by letting the atoms constitute a binary encoding of the state variable values. GT plan existence can be reduced to CPS plan existence by mapping each literal onto two distinct atoms, *ie.*, for any atom p in an instance of the GT problem, p and $\neg p$ are distinct atoms in the corresponding CPS instance. □

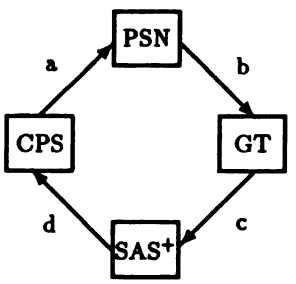


Figure 2: Reductions between the plan search problems in the four formalisms.

These reductions are not invariant wrt. to further restrictions, though. For example, negative preconditions do add to the expressiveness if there are no delete-lists.

The equivalence between SAS$^+$ planning and PSN planning also let us immediately conclude that unrestricted SAS$^+$ plan existence is PSPACE-complete since PSN plan existence is PSPACE-complete [Bylander, 1991a, Theorem 1].

[2]The equivalence of CPS and PSN plan existence can be implicitly derived from Bylander's results [1991a, Theorems 1 and 2] that both these problems are PSPACE-complete.

4 The SAS$^+$-PUS Planning Problem

4.1 The Problem

We have earlier presented some restricted planning problems: the SAS-PUBS problem [Bäckström and Klein, 1991b] and the SAS-PUS problem [Bäckström and Klein, 1991a]. These were defined using SAS-structures, which are more restricted than SAS$^+$-structures. In particular, both the initial and goal states were required to be total (all state variables defined). Furthermore, actions were not allowed to change a state variable from undefined to some defined value, but only from a defined value to some other defined value. The restriction to total goal states in the SAS-PUBS and SAS-PUS problems is a more serious restriction than it might appear. Trying to plan with partial goal states using a planner only allowing total goal states may lead to exponentiality in the worst case. Below we will define the SAS$^+$-PUS problem, which is the generalization of the SAS-PUS problem to SAS$^+$-structures, thus overcoming these problems.

The SAS$^+$-PUS planning problem is subject to three restrictions called *post-uniqueness*, *unariness* and *single-valuedness* (hence the acronym PUS). Post-uniqueness expresses that two different action types must not change the same state variable to the same value. Unariness expresses that every action type must change exactly one state variable. Single-valuedness, finally, expresses that all action types requiring the same state variable to remain constant during their occurrence must also require the *same* constant value. For example, single-valuedness prevents us from having two action types such that one requires a certain room to be lit during its occurrence while the other requires the same room to be dark.

Definition 4.1 Given a SAS$^+$ structure $\Phi = \langle \mathcal{M}, \mathcal{S}, \mathcal{H} \rangle$, a set of action types $\mathcal{H}' \subseteq \mathcal{H}$ is

- **post-unique** iff for all $h, h' \in \mathcal{H}'$, if $\mathrm{e}(h)[i] = \mathrm{e}(h')[i] \neq \mathrm{u}$ for some $i \in \mathcal{M}$, then $h = h'$;
- **unary** iff for all $h \in \mathcal{H}'$, there is some $i \in \mathcal{M}$ such that h affects i and for no $j \neq i$ does h affect j and
- **single-valued** iff there is some state $\hat{\mathrm{f}}_{\mathcal{H}'} \in \mathcal{S}$ such that $\mathrm{f}(h) \sqsubseteq \hat{\mathrm{f}}_{\mathcal{H}'}$ for all $h \in \mathcal{H}'$. □

For a somewhat more elaborate discussion of these restrictions, see Bäckström and Klein [1991a, 1991b] and Bäckström [1992]. Most other researchers [Bylander, 1991a, Erol *et al.*, 1992a] have only studied local restrictions on action types, like restricting

the number of pre-conditions or disallowing delete-conditions. Unariness is such a local restriction, while post-uniqueness and single-valuedness are global restrictions on the whole set of action types.

Definition 4.2 A SAS$^+$-structure $\Phi = \langle \mathcal{M}, \mathcal{S}, \mathcal{H} \rangle$ is a **SAS$^+$-PUS-structure** iff $\mathcal{H}$ is unary, post-unique and single-valued. The **SAS$^+$-PUS planning problem** is the restriction of the SAS$^+$ planning problem to SAS$^+$-PUS structures. □

4.2 Example: The Cappucino Brewer

In order to give some intuition for the restrictions on the SAS$^+$-PUS problem we present a small example that can be modelled as a SAS$^+$-PUS problem. The example is based on a very simple idea of how a Cappucino brewer *could* work. This example is chosen for illustrative purpose only and, as we will see below, the SAS$^+$-PUS formalism is not sufficiently expressive to model this example adequately. The brewer (see Figure 3) consists of a steam container, a coffee supply, a coffee filter and a milk steamer. In order to brew a

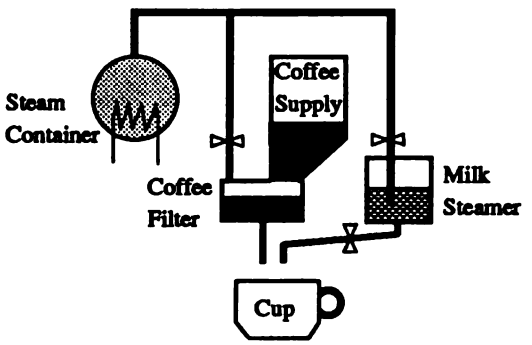


Figure 3: A simplified cappucino brewer

cup of espresso there must be coffee in the filter and steam in the steam container. If the milk in the milk steamer has been steamed to foam we might also top the espresso with milk foam, thus getting a cappucino (ignoring sugar and cocoa powder). We assume there is always coffee in the coffee supply and always milk or milk foam in the milk steamer. The state variables and their domains are described in Table 4.

Index	Description	Domain
1	Steam Pressure	{*High*, *Low*}
2	Coffee in filter	{*True*, *False*}
3	Cup content	{*Nothing*, *Espresso*, *Cappucino*}
4	Milk steamed	{*True*, *False*}

Table 4: State variables for the cappucino brewer

States are written as four-tuples of state variable values. For instance,

$$\langle \mathsf{H}, \mathsf{T}, \mathsf{u}, \mathsf{F} \rangle$$

denotes the state where the steam pressure is high (*ie.*, sufficiently high for operation), there is coffee in the filter, we do not know what is in the cup (or if there is anything at all in it) and there is milk, not foam, in the milk steamer. The available action types are described in Table 5. We have assumed that a heat-steam action heats the steam (or water) in the steam container until reaching operational pressure, whatever pressure (or water temperature) it starts from. Similarly, a steam-milk action can be executed even if there is already foam in the milk steamer. A dispense action dispenses coffee into the coffee filter. A brew action brews espresso into the cup if the steam pressure is sufficient and there is coffee in the filter. Finally, a fill-foam action fills the cup with milk foam.

action type	b	e	f
heat-steam	$\langle \mathsf{u}, \mathsf{u}, \mathsf{u}, \mathsf{u} \rangle$	$\langle \mathsf{H}, \mathsf{u}, \mathsf{u}, \mathsf{u} \rangle$	$\langle \mathsf{u}, \mathsf{u}, \mathsf{u}, \mathsf{u} \rangle$
dispense	$\langle \mathsf{u}, \mathsf{F}, \mathsf{u}, \mathsf{u} \rangle$	$\langle \mathsf{u}, \mathsf{T}, \mathsf{u}, \mathsf{u} \rangle$	$\langle \mathsf{u}, \mathsf{u}, \mathsf{u}, \mathsf{u} \rangle$
brew	$\langle \mathsf{u}, \mathsf{u}, \mathsf{N}, \mathsf{u} \rangle$	$\langle \mathsf{u}, \mathsf{u}, \mathsf{E}, \mathsf{u} \rangle$	$\langle \mathsf{H}, \mathsf{T}, \mathsf{u}, \mathsf{u} \rangle$
steam-milk	$\langle \mathsf{u}, \mathsf{u}, \mathsf{u}, \mathsf{u} \rangle$	$\langle \mathsf{u}, \mathsf{u}, \mathsf{u}, \mathsf{T} \rangle$	$\langle \mathsf{H}, \mathsf{u}, \mathsf{u}, \mathsf{u} \rangle$
fill-foam	$\langle \mathsf{u}, \mathsf{u}, \mathsf{E}, \mathsf{u} \rangle$	$\langle \mathsf{u}, \mathsf{u}, \mathsf{C}, \mathsf{u} \rangle$	$\langle \mathsf{u}, \mathsf{u}, \mathsf{u}, \mathsf{T} \rangle$

Table 5: Action types for the cappucino brewer

We assume the initial state is

$$s_0 = \langle \mathsf{u}, \mathsf{F}, \mathsf{N}, \mathsf{u} \rangle,$$

that is, we know nothing about the steam pressure and we do not know if there is milk or foam in the milk steamer. We do know, though, that both the coffee filter and the cup are empty. To brew an espresso we set the goal

$$s_E = \langle \mathsf{u}, \mathsf{u}, \mathsf{E}, \mathsf{u} \rangle$$

and to make a cappucino we set the goal

$$s_C = \langle \mathsf{u}, \mathsf{u}, \mathsf{C}, \mathsf{u} \rangle.$$

In both cases we only specify the content of the cup and leave all other state variables undefined ('don't care')—that is, both s_E and s_C are partial goal states.

A plan for making a cup of cappucino might look as in Figure 4. This plan has only one action of each type so we use the action-type names also as names for the actions. Both the brew-coffee action and the steam-milk action must be preceded by the heat-steam action to make certain that the steam pressure is operational. Coffee must be dispeused into the filter before brewing and we must ensure that the milk is steamed to foam before releasing it into the cup. Finally, the coffee must be brewed (released into the cup) before releasing the milk foam into the cup, otherwise the result will not be a cappucino—but more like a café au lait.

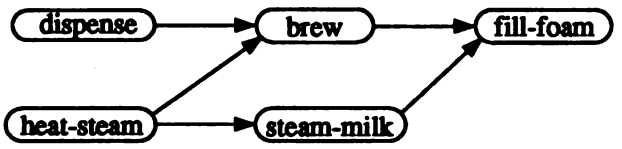


Figure 4: A plan for making a cup of cappucino

This is hardly an adequate modelling of the cappucino brewer. For example, we cannot model both that the coffee in the filter becomes used and that coffee is released into the cup when brewing; modelling this would require non-unary actions. Another problem—perhaps not so desperately needed in this example—is that we could not require the cup to be at different positions when filling it with coffee and milk foam respectively. On the other hand, post-uniqueness does not seem to prevent us from modelling the brewer.

Although the restrictions on the SAS$^+$-PUS problem seem very severe, they still allow for some interesting problems to be modelled (see Section 5.3 where a variant of the blocks world example from Section 2.2 is modelled).

5 Solving the SAS$^+$ Problem

5.1 Intended Minimal Plans

Later on in this section we will present an algorithm that finds minimal plans solving instances of the SAS$^+$-PUS problem. In order to understand the algorithm we must first introduce the concepts *i-chain* and *intended minimal plan* (IMP). The IMPs for an instance of the SAS$^+$ problem correspond exactly to the minimal plans for this instance. Hence, the definition of IMPs can be used as the specification for the planning algorithm.

Definition 5.1 An action sequence $\bar{\alpha} = \langle a_1, \ldots, a_n \rangle$ is an i-chain from s to t iff (1) $\mathrm{e}(a_k)[i] \neq \mathrm{u}$ for $1 \leq k \leq n$, (2) $\mathrm{b}(a_1)[i] \sqsubseteq s[i]$, (3) $t[i] \sqsubseteq \mathrm{e}(a_n)[i]$ and (4) $\mathrm{b}(a_k)[i] \sqsubseteq \mathrm{e}(a_{k-1})$ for $1 < k \leq n$. If $t[i] \sqsubseteq s[i]$, we also regard $\langle\rangle$ as an i-chain from s to t. □

Since a single-valued set of action types can only specify one defined value for each state variable to be defined in the prevail-conditions of the action types, it is practical to insert a shorthand to denote these unique values. We define

$$\hat{\mathrm{f}}_{\mathcal{H}} = \sqcup_{h \in \mathcal{H}} \mathrm{f}(h),$$

which is well-defined since $\mathcal{H}$ is single-valued.

Definition 5.2 Given a SAS$^+$-PUS planning problem $\Pi = \langle \langle \mathcal{M}, \mathcal{S}, \mathcal{H} \rangle, s_0, s_\star \rangle$, a plan $\Delta = \langle \mathcal{A}, \prec \rangle$ over Π is **an intended minimal plan** (IMP) for Π iff there are sets $\mathcal{F}_1, \ldots, \mathcal{F}_{|\mathcal{M}|} \subseteq \mathcal{A}$ such that

1. for each $i \in \mathcal{M}$, there is an i-chain $\bar{\gamma}_i$ in Δ from s_0 to $s_\star$;
2. for each $a \in \mathcal{A}$ and for each $i \in \mathcal{M}$, $a \in \mathcal{F}_i$ iff $\mathrm{f}(a)[i] \neq \mathrm{u}$;
3. for each $i \in \mathcal{M}$, if $\mathcal{F}_i \neq \emptyset$, then there is a minimal i-chain $\bar{\alpha}_i$ in Δ from s_0 to $\hat{\mathrm{f}}_{\mathcal{H}}$ such that $\bar{\alpha}_i \prec a$ for all $a \in \mathcal{F}_i$;
4. for each $i \in \mathcal{M}$, if $\mathcal{F}_i \neq \emptyset$, then there is a minimal i-chain $\bar{\beta}_i$ in Δ from $\hat{\mathrm{f}}_{\mathcal{H}}$ to $s_\star$ such that $a \prec \bar{\beta}_i$ for all $a \in \mathcal{F}_i$;
5. $\mathcal{A}$ is minimal wrt. the above and
6. $\prec$ is a minimal strict partial order on $\mathcal{A}$ satisfying the above. □

We will see later in this section that any minimal plan solving some instance of the SAS$^+$-PUS problem is isomorphic to the IMPs for this instance. Hence, any minimal SAS$^+$-PUS plan (or IMP) can be understood as follows.

For each $i \in \mathcal{M}$ there is at most one main goal, $s_\star[i]$, and one subgoal, $\hat{\mathrm{f}}_{\mathcal{H}}[i]$. Furthermore, there are three i-chains: $\bar{\gamma}_i$, $\bar{\alpha}_i$ and $\bar{\beta}_i$. The i-chain $\bar{\gamma}_i$ achieves the main goal $s_\star[i]$ from the initial state $s_0[i]$, the i-chain $\bar{\alpha}_i$ achieves the subgoal $\hat{\mathrm{f}}_{\mathcal{H}}[i]$ from the initial state and the i-chain $\bar{\beta}_i$ achieves the main goal from $\hat{\mathrm{f}}_{\mathcal{H}}[i]$. Except for the case where $\mathrm{f}(a)[i] = \mathrm{u}$ for all actions a in the plan, $\bar{\gamma}_i$ is not strictly needed since $\bar{\gamma}_i = (\bar{\alpha}_i; \bar{\beta}_i)$ in this case. As an example, consider a domain $\mathcal{S}_i = \{1, 2, 3, 4\}$ and let $s_0[i] = 1$, $s_\star[i] = 3$ and $\hat{\mathrm{f}}_{\mathcal{H}}[i] = 4$. The actions of the plan that affect i can then be viewed as in Figure 5. The various i-chains for i are then as follows:

$$\begin{aligned} \bar{\alpha}_i &= \langle a_1, a_2, a_3 \rangle, \\ \bar{\beta}_i &= \langle a_4, a_5, a_6 \rangle, \\ \bar{\gamma}_i &= (\bar{\alpha}_i; \bar{\beta}_i) = \langle a_1, \ldots, a_6 \rangle. \end{aligned}$$

This is a worst-case scenario in the sense that no minimal plan for this instance have more actions affecting i. Note, for instance, that $\bar{\alpha}_i$ passes through $s_\star[i]$ and that $\bar{\beta}_i$ cannot go directly from $\hat{\mathrm{f}}_{\mathcal{H}}[i]$ to $s_\star[i]$ without passing $s_0[i]$. Both these phenomena are caused by the post-uniqueness.

If we instead let $\hat{\mathrm{f}}_{\mathcal{H}} = 2$ and $s_\star[i] = 4$, we have the following i-chains:

$$\begin{aligned} \bar{\alpha}_i &= \langle a_1 \rangle, \\ \bar{\beta}_i &= \langle a_2, a_3 \rangle, \\ \bar{\gamma}_i &= (\bar{\alpha}_i; \bar{\beta}_i) = \langle a_1, a_2, a_3 \rangle. \end{aligned}$$

Furthermore, if $\hat{\mathrm{f}}_{\mathcal{H}} = s_0[i]$ or $\hat{\mathrm{f}}_{\mathcal{H}} = s_\star[i]$, then $\bar{\alpha}_i = \langle\rangle$ or $\bar{\beta}_i = \langle\rangle$ respectively. In the case where $s_0[i] = s_\star[i] \neq \hat{\mathrm{f}}_{\mathcal{H}}$ we have two i-chains from $s_0[i]$ to $s_\star[i]$, namely $\langle\rangle$ and $(\bar{\alpha}_i; \bar{\beta}_i)$. Either of these is a possible choice for

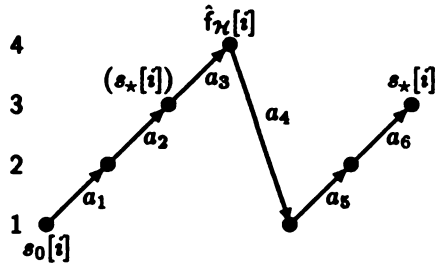


Figure 5: The actions affecting an index i in an IMP in the worst case.

$\bar{\gamma}_i$, but it must be one of these in order to satisfy the minimality requirement in Definition 5.2.

An interesting property of the IMPs and, hence, of the minimal SAS$^+$-PUS plans is that each prevail-condition need only be achieved one, that is, each $\bar{\alpha}_i$ serves all actions a in the plan having a defined prevail-condition for i. The set $\mathcal{F}_i$ collects together all such actions for each $i \in \mathcal{M}$.

As a consequence of the properties discussed above, we make the interesting observation that no IMP contains more than two actions of each type.

As we claimed above, it can be proven that the IMPs correspond exactly to the minimal plans for instances of the SAS$^+$-PUS problem.

Theorem 5.3 Given an instance Π of the SAS$^+$-PUS problem, any IMP for Π is a minimal plan solving Π and if Π has a solution, then there exists some IMP for Π. □

Proof sketch: Let Π be an instance of the SAS$^+$ problem and also suppose Δ is an IMP for Π. For every $\bar{\alpha} \in CS(\Delta)$, we prove by induction over the length of $\bar{\alpha}$ that it is admissible wrt. s_0, so $ACS(s_0, \Delta) = CS(\Delta)$. Then prove $\mathit{result}(s_0, \bar{\alpha}) = s_\star$ for all $\bar{\alpha} \in CS(\Delta)$. Hence, Δ is a plan solving Π.

For the converse case, suppose Δ is a plan solving Π. We can prove that there is some plan Δ' solving Π that does not have any mutually unordered actions with the same effect. Further prove that there exist some subset of the actions in Δ' that constitute and IMP for Π.

Finally we can prove by contradiction that any IMP for Π is a minimal plan solving Π since we know from above that every minimal plan contains some IMP.

The detailed proof of the above theorem appears in [Bäckström, 1992, Appendix B].

5.2 Algorithm

Algorithm 5.1 finds minimal plans solving SAS$^+$ instances.

finds a minimal plan whenever there is one

Algorithm 5.1

```
1  procedure BuildChain(H, s_F, s_T, i, A, T, R, U);
2  begin
3    if s_T[i] ⊑ s_F[i] then return ⟨⟩;
4    else
5      Find h ∈ H such that h[i] = s_T and instantiate as a;
6      if no such a or type(a) ∈ U then fail;
7      else
8        C ← BuildChain(H, s_F, b(a), i, A, T,
                        R, (U ∪ {type(a)}));
9        A ← A ∪ {a}; T ← T ∪ {a};
10       if C ≠ ⟨⟩ then
11         Order Last(C) before a in R;
12       return (C;a);
13 end BuildChain;

1  procedure Plan(M, H, s_0, s_⋆);
2  begin
3    A ← ∅; T ← ∅; R ← ∅; F_i ← ∅;
     γ̄_i ← ⟨⟩; ᾱ_i ← ⟨⟩; β̄_i ← ⟨⟩;
4    for i ∈ M loop
5      γ̄_i ← BuildChain(H, s_0, s_⋆, i, A, T, R, ∅);
6    while T ≠ ∅ loop
7      Select arbitrary a ∈ T and let T ← T − {a};
8      for i ∈ M loop
9        if f(a)[i] ≠ u then
10         if F_i = ∅ then
11           if γ̄_i = ⟨a_1, ..., a_n⟩ such that 1 ≤ k ≤
     n and
                    f(a) ⊑ e(a_k) then
12             ᾱ_i ← ⟨a_1, ..., a_k⟩;
13             β̄_i ← ⟨a_{k+1}, ..., a_n⟩;
14           else ᾱ_i ← BuildChain(H, s_0, f(a), i,
                                   A, T, R, ∅);
15           if s_⋆[i] = u then β̄_i ← ⟨⟩;
16           else β̄_i ← (BuildChain(H, f(a), s_0,
                         i, A, T, R, ∅); γ̄_i);
17             if β̄' ≠ ⟨⟩ and γ̄_i ≠ ⟨⟩ then
18               Order Last(β̄') before First(γ̄_i)
                 in R;
19         F_i ← F_i ∪ {a};
20         if ᾱ_i ≠ ⟨⟩ then
21           Order Last(ᾱ_i) before a in R;
22         if β̄_i ≠ ⟨⟩ then
23           Order a before First(β̄_i) in R;
24   if R is not irreflexive then fail;
25   else return ⟨A, R⟩;
26 end Plan;
```

The algorithm works by implementing Definition 5.2. Procedure *BuildChain* constructs a minimal i-chain

from s_F to s_T. The main procedure, *Plan*, works somewhat similar to the GPS algorithm [Newell and Simon, 1963], but it works on several goals simultaneously and when working on a goal, it does not continue with all subgoals of this goal before continuing with other goals.

We assume below that *Plan* is called with $M = \mathcal{M}$, $H = \mathcal{H}$ and s_0 and $s_\star$ as given in some instance $\Pi = \langle\langle \mathcal{M}, \mathcal{S}, \mathcal{H}\rangle, s_0, s_\star\rangle$ of the SAS$^+$-PUS problem.

To understand how the algorithm works one must first understand how *BuildChain* works. Procedure *BuildChain* takes two states, s_F and s_T, and a state variable index, i, as main parameters. Its purpose is to construct an i-chain from s_F to s_T using *new* actions of types in H but not in U. If this succeeds, then *BuildChain* returns such an i-chain $\bar{\alpha}$ and, as a side effect, adds the actions in $\bar{\alpha}$ to the sets A and T and adds its chain order $\prec_{\bar{\alpha}}$ to R. If there is no such i-chain, then *BuildChain* fails, *ie.*, the whole algorithm reports failure. *BuildChain* constructs the i-chains recursively from the end in a straightforward way. The only subtle detail is the use of the set U, which is used to avoid looping. U is only used when *BuildChain* calls itself recursively and, hence, is always the empty set when *Plan* calls *BuildChain*. Whenever *BuildChain* instantiates some action type h it adds h to U before the next recursive call. Hence, if an action of type h is needed also in a subsequent recursive call, then this can be detected by inspection of U. If this is the case, then the i-chain constructed so far defines a loop in the state space so, because of post-uniqueness, there cannot be any i-chain from s_F to s_T. If this situation was not detected, *BuildChain* would continue generating an infinite number of duplicates of this action sequence, defining an infinite loop.

Procedure *Plan* follows Definition 5.2 closely. The variables are used as follows. The set A is used to accumulate all actions generated and will finally correspond to the set of actions in the final plan, if there is one. The purpose of the set T will become evident below. The relation R is used to accumulate temporal constraints between the actions and will finally correspond to the plan order, except that R will not contain any transitive 'arcs'. The F_i sets correspond to the $\mathcal{F}_i$ sets in Definition 5.2. Finally, for each $i \in \mathcal{M}$, $\bar{\gamma}_i$, $\bar{\alpha}_i$ and $\bar{\beta}_i$ will contain the i-chains dictated by requirements (1), (3) and (4) respectively in Definition 5.2.

The loop in lines 4–5 implements requirement (1) and stores the resulting i-chains in the $\bar{\gamma}_i$ variables. The while loop in lines 6–23 is the heart of the algorithm and implements requirements (2)–(4). It works as follows. *BuildChain* assures that every action ever inserted into A is also inserted into T, even when *BuildChain* is called from within the body of this while loop. Hence, every action a ever inserted into A (and, thus, into T) is eventually removed from T and processed by the body of this while loop in the following way. For each $i \in \mathcal{M}$, *Plan* tests whether $\mathsf{f}(a)[i] \neq \mathsf{u}$. If this is not the case, then requirements (3) and (4) are trivially satisfied. Otherwise, lines 10–19 assure that there exist i-chains $\bar{\alpha}_i$ and $\bar{\beta}_i$ as required by (3) and (4) respectively and that a is added to F_i. Lines 20–23 assure that a is ordered wrt. to $\bar{\alpha}_i$ and $\bar{\beta}_i$ as required by (3) and (4). The non-trivial part occurs in lines 10–19. The test in line 10 assures that lines 11–18 are executed at most once for each $i \in \mathcal{M}$ and that this happens for the first action removed from T that has a defined pre-condition for i. Line 11 tests whether the i-chain $\bar{\gamma}_i$ can be split into two i-chains that satisfy requirements (3) and (4). If this is the case, then $\bar{\gamma}_i$ is split and the resulting i-chains are stored in $\bar{\alpha}_i$ and $\bar{\beta}_i$—hence, no new actions need be introduced. Otherwise, *BuildChain* is called twice to construct new such i-chains. However, if $s_\star[i] = \mathsf{u}$, then it constructs the empty i-chain. It does not construct an entirely new i-chain from $\mathsf{f}(a)$ to $s_\star$ in the other case either. Instead, it constructs a new i-chain from $\mathsf{f}(a)$ to s_0 and concatenates this with $\bar{\gamma}_i$ to form $\bar{\beta}_i$. Since the actions in $\bar{\gamma}_i$ are 'reused' this is actually a minimal i-chain satisfying requirement (4). Finally, line 24 tests whether R is irreflexive, thus assuring that R^+ is a partial order.

Theorem 5.4 If there is a plan solving an instance Π of the SAS$^+$-PUS problem, then Algorithm 5.1 returns a tuple $\langle A, R\rangle$ such that $\langle A, R^+\rangle$ is a minimal plan solving Π and otherwise it fails. □

The correctness of this theorem should appear plausible from the discussion above and the proof appears in Bäckström [1992, Chapter 4].

The algorithm does not compute the transitive closure of R since this operation would dominate and increase the complexity figure. However, most applications are not likely to require the transitive closure anyway and it could easily be computed if needed. Furthermore, the plans found by the algorithm are maximally parallel [Bäckström and Klein, 1991a, 1991b].

The algorithm is proven sound and complete.

Theorem 5.5 Algorithm 5.1 has a worst case time complexity of $O(m^2n^2)$ time, where m is the number of state variables and n is the maximum domain size (number of values) of any state variable. □

Proof: We first analyse procedure *BuildChain*. Each chain of recursive calls of *BuildChain* instantiates only action types affecting one index, i, and no action type is instantiated more than once, which the variable U is used to keep track of. Hence, post-uniqueness guarantees that there are at most n consecutive recursive calls of *BuildChain*, each taking at most $O(h)$ time. Hence, *BuildChain* takes $O(hn)$ time, where h is the number of action types in H.

We can now analyse procedure *Plan*. We first make the observation that $|A| \in O(h)$ and $|R| \in O(h^2)$ since

no IMP contains more than two actions of each type. It is also reasonable to assume that $h \in \Theta(mn)$ since it would otherwise not be a concise encoding [Garey and Johnson, 1979].

We assume that R is implemented as an adjacency matrix so the variables can be initialized in $O(h^2)$ time. The loop in lines 4–5 runs in O(mhn) time. We note that lines 11–18 runs at most m times and take $O(hn)$ time, giving a total execution time of $O(mhn)$. Lines 20–23 take $O(1)$ time so the whole while loop takes $O(mhn)$ time. R can be tested for irreflexivity in $O(hm)$ time by topological sorting. Obviously, *Plan* runs in $O(mhn)$ time, which simplifies to $O(m^2n^2)$. □

A more elaborate proof appears in [Bäckström, 1992, Chapter 4].

Although this algorithm solves a more general problem than the previous SAS-PUS algorithm it, surprisingly, exhibits a better complexity figure (the SAS$^+$-PUS algorithm subsumes the SAS-PUS algorithm, of course). The main motivation for tractability is that the P, U and S restrictions impose sufficiently much structure on the problem so that search can be avoided. Furthermore, the algorithm can even be modified to run in $O(m^2n)$ time [Bäckström, 1992, Chapter 4].

An analysis of the complexity of planning for other subproblems of the SAS$^+$ problem appears in Chapter 5 of Bäckström [1992].

5.3 Blocks World in SAS$^+$-PUS

Although the SAS$^+$-PUS problem may seem very restricted, it is sufficiently expressive to solve a variant of the blocks-world problem. Gupta and Nau [1991] defined the *elementary blocks world problem* (EBW) and proved that finding minimal plans for this problem is NP-hard. The restricted problem, EBW$^-$, where blocks may not be moved directly from one stack to another is not quite so hard, however. Bylander [1991a] has shown that the plan existence problem is tractable for EBW$^-$. Moreover, the EBW$^-$ problem can be encoded as an instance of the SAS$^+$-PUS problem (using the same technique as Bylander) so we can even *find minimal* plans for instances of EBW$^-$ in polynomial time. Furthermore, any plan solving an instance of EBW$^-$ also solves the corresponding instance of EBW and is at most twice as long. Hence, the SAS$^+$-PUS algorithm can be used as an approximation algorithm for EBW producing plans that are most twice the length of a minimal plan. Finding such an approximation takes $O(n^4)$ time where n is the number of blocks. This is further discussed in Bäckström [1992, Ch 5].

6 Conclusions

We have presented the SAS$^+$ planning formalism, which overcomes some serious problems in our previous SAS formalism. We have shown that, contrary to our intuition, the SAS$^+$ formalism is equally expressive as some 'standard' propositional formalisms, like ground TWEAK. However, the SAS$^+$ formalism has conceptual advantages over traditional STRIPS formalisms for some application domains. Furthermore, the SAS$^+$ formalism made it possible to identify certain restrictions that give tractability and are natural to express in this formalism, but that appear very unnatural if expressed in the traditional formalisms [Bäckström, 1992, Chapter 5]. Finally, we have presented an algorithm that finds minimal plans in polynomial time for a restricted version of the SAS$^+$ planning problem, the SAS$^+$-PUS problem. Hence, the tractability result for SAS-PUS planning carries over also to the more expressive SAS$^+$ formalism and, surprisingly, the new algorithm even exhibits a better complexity figure.

Acknowledgements

The work on SAS$^+$ planning build on previous research together with Inger Klein. Bernhard Nebel and Bart Selman provided helpful comments on earlier versions of the material underlying this paper. The research was done under STUF grant 90-358.

References

[AAAI-92, 1992] *Proceedings of the Tenth (US) National Conference on Artificial Intelligence (AAAI-92)*, San José, CA, USA, July 1992.

[Åström *et al.*, 1991] Karl-Johan Åström, Albert Benveniste, et al. Facing the challenge of computer science in the industrial applications of control: a joint IEEE CSS–IFAC project. Progress Report, March 1991.

[Bacchus and Yang, 1992] Fahiem Bacchus and Qiang Yang. The expected value of hierarchical problem-solving. In AAAI-92 [1992], pages 369–374.

[Bäckström and Klein, 1991a] Christer Bäckström and Inger Klein. Parallel non-binary planning in polynomial time. In Reiter and Mylopoulos [1991], pages 268–273.

[Bäckström and Klein, 1991b] Christer Bäckström and Inger Klein. Planning in polynomial time: The SAS-PUBS class. *Computational Intelligence*, 7(3):181–197, August 1991.

[Bäckström and Nebel, 1992] Christer Bäckström and Bernhard Nebel. On the computational complexity of planning and story understanding. In Bernd

Neumann, editor, *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, Vienna, Austria, August 1992. Wiley.

[Bäckström, 1988a] Christer Bäckström. Reasoning about interdependent actions. Licentiate Thesis 139, Department of Computer and Information Science, Linköping University, Linköping, Sweden, June 1988.

[Bäckström, 1988b] Christer Bäckström. A representation of coordinated actions characterized by interval valued conditions. In Zbigniew W Ras and Lorenza Saitta, editors, *Proceedings of the Third International Symposium on Methodologies for Intelligent systems (ISMIS-88)*, pages 220–229, Torino, Italy, October 1988. North-Holland.

[Bäckström, 1992] Christer Bäckström. *Computational Complexity of Reasoning about Plans.* PhD thesis, Linköping University, Linköping, Sweden, June 1992.

[Brown, 1987] Frank Brown, editor. *The Frame Problem in Artificial Intelligence, Proceedings of the 1987 Workshop*, Lawrence, KS, USA, April 1987. Morgan Kaufman.

[Bylander, 1991a] Tom Bylander. Complexity results for planning. In Reiter and Mylopoulos [1991], pages 274–279.

[Bylander, 1991b] Tom Bylander. Tractability and artificial intelligence. *Journal of Experimental and Theoretical AI*, 3:171–178, 1991.

[Chapman, 1987] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[Dean and Boddy, 1988] Thomas Dean and Mark Boddy. Reasoning about partially ordered events. *Artificial Intelligence*, 36:375–399, 1988.

[Erol *et al.*, 1992a] Kutluhan Erol, Dana S Nau, and V S Subrahmanian. On the complexity of domain-independent planning. In AAAI-92 [1992], pages 381–386.

[Erol *et al.*, 1992b] Kutluhan Erol, Dana S Nau, and V S Subrahmanian. When is planning decidable? In *Artificial Intelligence Planning Systems: Proceedings of the First International Conference*, pages 222–227, College Park, MD, USA, June 1992. Morgan Kaufman.

[Fikes and Nilsson, 1971] Richard E Fikes and Nils J Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Garey and Johnson, 1979] Michael Garey and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York, 1979.

[Golumbic, 1992] Martin Charles Golumbic. Algorithms and complexity for reasoning about time. In AAAI-92 [1992], pages 741–747.

[Gupta and Nau, 1991] Naresh Gupta and Dana S Nau. Complexity results for blocks-world planning. In *Proceedings of the Ninth (US) National Conference on Artificial Intelligence (AAAI-91)*, pages 629–633, Anaheim, CA, USA, July 1991. AAAI Press/MIT Press.

[Hayes, 1981] Patrick J Hayes. The frame problem and related problems in artificial intelligence. In Bonnie Lynn Webber and Nils J Nilsson, editors, *Readings in Artificial Intelligence*, pages 223–230. Morgan Kaufman, 1981.

[Klein and Bäckström, 1991] Inger Klein and Christer Bäckström. On the planning problem in sequential control. In *Proceedings of the 30th IEEE Conference on Decision and Control*, Brighton, UK, December 1991.

[Knoblock, 1990] Craig A Knoblock. A theory of abstraction for hierarchical planning. In D Paul Benjamin, editor, *Change of Representation and Inductive Bias, Firts International Workshop. Revised selected papers.*, pages 81–104, Boston, MA, USA, 1990. Kluwer.

[Korf, 1987] Richard E Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88, 1987.

[Nebel and Bäckström, 1992] Bernhard Nebel and Christer Bäckström. On the computational complexity of temporal projection and plan validation. In AAAI-92 [1992], pages 748–753.

[Newell and Simon, 1963] Allen Newell and Herbert A Simon. GPS, a program that simulates human thought. In Feigenbaum and Feldman, editors, *Computers and Thought.* McGraw Hill, 1963.

[Passino and Antsaklis, 1989] K M Passino and P J Antsaklis. A system and control theoretic perspective on artificial intelligence planning systems. *Applied Artificial Intelligence*, (3):1–32, 1989.

[Reiter and Mylopoulos, 1991] Ray Reiter and John Mylopoulos, editors. *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, Sydney, Australia, August 1991. Morgan Kaufman.

[Sandewall and Rönnquist, 1986] Erik Sandewall and Ralph Rönnquist. A representation of action structures. In *Proceedings of the Fifth (US) National Conference on Artificial Intelligence (AAAI-86)*, pages 89–97, Philadelphia, PA, USA, August 1986. Morgan Kaufman.

[Sandewall, 1992] Erik Sandewall. Features and fluents. Unpublished book manuscript, Dept. of Computer and Information Science, Linköping University, January 1992.

II.

Specialized Reasoning

Stepwise-Decomposable Influence Diagrams

(Nevin) Lianwen Zhang and David Poole
Department of Computer Science
University of British Columbia
Vancouver, B.C., V6T 1Z2, Canada
E-mail: lzhang@cs.ubc.ca poole@cs.ubc.ca

Abstract

In the literature, authors mostly talk about influence diagrams that are regular, no-forgetting and having one value node. In this paper, a less restrictive concept of influence diagrams, i.e, the concept of stepwise-decomposable influence diagrams is introduced and studied.

As a knowledge representation scheme, stepwise-decomposable influence diagrams are more natural than no-forgetting influence diagrams not only because they are less restrictive, but also because they corresponds exactly to stepwise-solvability.

Stepwise-decomposable influence diagrams are advantageous for the following reasons. Firstly, they enable the representation of knowledge about a piece of information being irrelevant to a decision, and consequently enable us to prevent the decision tables from being unnecessarily large at the problem formulation stage. Secondly, stepwise-decomposability allows us to clearly establish the relationship between influence diagram evaluation and Bayesian net computations. Thirdly, because of the relationship, we have the freedom to evaluate an influence diagrams in a number of ways, while Shachter's arc reversal and node reduction algorithm is only one of them, and not necessarily the most efficient one. Finally, stepwise-decomposable influence diagrams provide a convenient setting for investigating when an arc into a decision node can be harmlessly removed by a preprocessing step. This enables us to prevent, in the evaluation process the decision tables from being unnecessarily large.

On a more abstract level, stepwise-decomposable influence diagram also lead to a simple and insightful mathematical theory of influence diagrams, particularly efficient and easy to understand algorithms.

1 INTRODUCTION

Influence diagrams were introduced by Miller *et al* (1976) and Howard and Matheson (1984) as a graphical representation of decision problems. In comparison with decision trees (Raiffa 1968), influence diagrams are more intuitive and can be easily understood by people in all walks of life and degrees of proficiency. They are equally precise and more structured, thus allow more efficient treatment by computers.

An influence diagram is an acyclic directed graph with three types of nodes: chance nodes, decision nodes, and a single value node. They respectively represent random quantities, decisions to make and the utilities of alternatives.

In a one decision maker situation, decisions are made sequentially. So, if an influence diagram is a representation of a single decision maker's view of the world, then it is reasonable to require it to be *regular*, that is, all the decision nodes are ordered (Howard and Matheson 1984 and Shachter 1986).

Arcs into a decision node are interpreted as indications of information availability (Howard and Matheson 1984). More specifically, an arc from a node a to a decision node d means that the value of (the variable represented by) a is known at the time the decision d is to be made. The lack of an arc from a to d means that information about a is not available at the time when d is to be made. Thus, if an influence diagram represents a single decision maker who does not forget information, any decision node and its parents must be parents of subsequent decision nodes (Howard and Matheson 1984). If it is the case, the diagram is said to be *no-forgetting* (Shachter 1986).

A drawback of interpreting arcs into decision nodes as indications of information availability is that knowledge about a node being irrelevant to a decision can

not be represented. Information about a node a may be irrelevant to a decision d even if it is available at the time the decision d is to be made. See section 2 for an example.

This inability to represent irrelevancies results in decision tables of sizes exponential in the number of decision nodes and their parents. Suppose there are 10 decision nodes, and every decision node have three chance node parents that they share with only subsequent decision nodes. Also suppose all the nodes have two possible values. Then the size of the decision table for the last decision nodes is 2^{39}.

In this paper, we interpret arcs into decision nodes as indications of potential functional dependency. More specifically, the existence of an arc from a to d means that the decision d potentially depends on the value of a, while the lack of an arc from a to d implies that information about a is irrelevant to the decision d. Under this interpretation, it is no longer reasonable to require that any decision node and its parents be parents of all the subsequent decision nodes. In other words, the no-forgetting requirement has to be lifted.

We also allow influence diagrams to represent situations where more than one person is making the decisions. As a consequence, the regularity requirement has to be lifted, for there may be groups of decisions which can be made in any relative orders. See section 2 for an example.

The one-value requirement is also lifted in this paper for technical convenience and for simpler and more efficient algorithms.

Lifting of the one-value-node requirement, regularity, and the no-forgetting requirement leads to a very general concept of influence diagrams (section 2).

Usually influence diagrams in the most general sense are not stepwise-solvable. That is, in general, they can not be evaluated step by step by considering one decision at a time. One has to consider potentially all the decisions at the same time. The complexity of doing so may be prohibitive. For this reason, we shall impose a new requirement called stepwise-decomposability (section 5) on influence diagrams.

Even though the ability to represent knowledge about a node being irrelevant to a decision is an important motivation for studying stepwise-decomposable influence diagrams, it is not the most important one. The most important motivation for studying stepwise-decomposable influence diagrams is that they lead to a simpler and more insightful mathematical theory of influence diagrams. They also lead to simpler and more efficient algorithms.

The organization of the paper is as follows. Section 2 introduces a very general concept of influence diagrams and gives a brief exposition of its semantics. The computational problem of evaluating an influence diagram is defined in section 3, together with some other technical concepts. In section 4, an attempt is made to formalize the intuitive concept of stepwise-solvability. Stepwise-decomposable influence diagrams are introduced in section 5. In section 6, we show that evaluating stepwise-decomposable influence diagrams amounts to computing marginal probabilities in local Bayesian nets. It is also shown that, in a sense, stepwise-decomposability and stepwise-solvability are equivalent. In section 7, the issue as to when an arc into a decision node can be harmlessly removed by a preprocessing step is addressed. Conclusions are provided in section 8.

2 INFLUENCE DIAGRAMS: DEFINITION AND SEMANTICS

Let us begin by giving a definition of influence diagrams. Our definition is more general than all the definitions in the literature to the best ot our knowledge.

An *influence diagram* I is a triplet $I = (V, A, \mathcal{P})$, where

1. The first component V is a set of nodes, which partitions into a set of *chance nodes* C, a set of *decision nodes* D, and a set of *value nodes* U;
2. The second component A is a set of arcs in V, which, together with V, constitutes a connected acyclic directed graph $G = (V, A)$[1], in which all the value nodes are leaves; and
3. The third component $\mathcal{P}$ is a set of conditional probabilities $\{P(x|\pi(x)) : x \in C \cup U\}$, where $\pi(x)$ denotes the set of all the parents of x[2].

According to this definition, an influence diagram contains the conditional probabilities of all the chance nodes and value nodes, but not the decision nodes. The conditional probability of a value node v is obtained from a value function. More specifically, $P(v = v_0|\pi(v) = X_0)$ is 1 if v_0 is the expected value when the parents $\pi(v)$ of v assume the values X_0, and it is 0 otherwise (see the example below.).

Briefly, the semantics of an influence diagram are as follows. Decision nodes represent decisions to be made, chance nodes represent random variables that are relevant to the decision problem, and value nodes represents the utilities of alternatives.

[1] In this paper, well known graph theory terms such as connected acyclic directed graphs, parents, children, nondescendants, leaves, and roots will be used without giving the definitions. The reader is directed to Lauritzen *et al* (1990) for exact definitions.

[2] Note that when x is a root, $\pi(x)$ is empty. When it is the case, $P(x|\pi(x))$ stands for the prior probability of x.

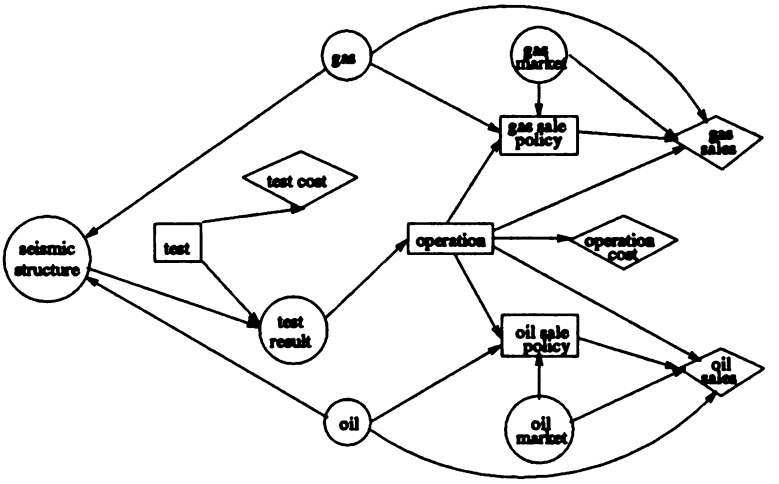


Figure 1: A representation of the oil company problem in terms of influence diagrams.

Arcs into chances nodes and value nodes represent probabilistic dependence. They are thus called *conditional* arcs. A value node or a chance node x depends on its parents $\pi(x)$ in the way as specified by the conditional probability $P(x|\pi(x))$. On the other hand, it is independent of all its nondescendants given all its parents.

Arcs into decision nodes denote potential functional dependence. They are thus called *decision dependency* arcs. The existence of an arc from a node a into a decision node d means that the decision d depends on information about a, in the sense that different values of a may result in different decisions about d even when the values of the other parents of d are fixed. On the other hand, the lack of an arc from a to d means that information about a is irrelevant to the decision d. That is, given the values of all the other parents of d, the value of a does not affect the decision d at all.

Note that our interpretation of arc into decision nodes is different from the one in the literature (Howard and Matheson 1984, Shachter 1986), where an arc from a node a to a decision node d was interpreted as an indication that information about a is available at the time when the decision d is to be made. Thus such arcs were called *informational* arcs. A drawback of this interpretation is that knowledge about a node being irrelevant to a decision can not be represented (see the following example).

In an influence diagram, decisions at decision nodes are made based on the values of their respective parents and are made to maximize the expected value of the sum of all the value nodes.

Figure 1 shows an example influence diagram. As a convention, decision nodes are drawn as rectangles, chance nodes as circles, and value nodes as diamonds. The story goes as follows.

An oil company is considering whether or not it should open up operations in a new area. The company will first decide whether or not to order a seismic structure test. Based on the test results, the company will decide whether or not to open the operation. When oil and gas are produced, the company will then decide its gas and oil sale policies according to market information and the quality of gas and oil produced. All those decisions are to be made to maximize the company's profit.

The root chance nodes are "gas", "oil", "gas-market" and "oil-market". The test results depend on the decision to test and the seismic structure, which in turn depends on quality and quantity of gas and oil underground.

If ordered, the test will cost the company some money. The operation itself has a cost too. The company can make a certain amount of money from selling gas as well as from selling oil. Gas (oil) sales depends on gas (oil) sale policy, the quantity of gas (oil) and the gas (oil) market. The company's profit is the sum of gas sales and oil sales minus test cost and operation cost.

To simplify the problem, we have assumed that the quality and quantity of gas and oil that will have been produced are the same as the quality and quantity of gas and oil underground. We have also assumed that costs in selling gas and oil are absorbed in gas and oil sales.

In this example, one normally will not concern oneself with the "test-result" when making the "gas-sale-policy", for at that time one knows the quantity and quality of gas produced, which, plus market information, are all what s/he needs to make the decision. Information about "test-result" is irrelevant to the de-

cision "gas-sale-policy", even though it may be available. Because we interpret arcs into decision nodes as indications of potential functional dependency, we are able to represent such a piece of knowledge by not drawing an arc from "test-result" to "gas-sale-policy".

Also in the example, it may be the case that "gas-sale-policy" is made by people in the "gas department", and "oil-sale-policy" is made by people in the "oil department", while "operation" is made by the executive board of the company. So, even though both the decision "gas-sale-policy" and the decision "oil-sale-policy" will be made after the decision "operation", they can be made in any relative order. This supports abandoning of regularity.

To end this section, we introduce two concepts that will be useful later. A *parametric influence diagram* is simply an influence diagram, except that some of its conditional probabilities contains *parameters* — variables of the problem that do not appear in the underlying graphical structure.

A *Bayesian net* is an influence diagram without decision and value nodes.

3 Computations in influence diagrams

In this section, we specify what we want to compute in an influence diagram.

Let $I = (V, A, \mathcal{P})$ be an influence diagram, where V partitions into C, D, and U. For any variable $x \in V$, let Ω_x denote the *frame* of x, i.e, the set of possible values of x. For each decision node $d \in D$, the decision maker is to decide upon a function, called the *decision function* for d, from the Cartesian product $\prod_{x\in\pi(d)} \Omega_x$ of the frames of the parents of d to the frame Ω_d of d itself. A collection of such functions, one function for each decision node, constitute a *policy* for the influence diagrams.

A function $f : \prod_{x\in\pi(d)} \Omega_x \rightarrow \Omega_d$ can be represented by the conditional probability $P_f(d|\pi(d))$ defined as follows:

$$P_f(d = d_0|\pi(d) = X_0) = \begin{cases} 1 & \text{if } f(X_0) = d_0 \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where d_0 stands for a value of d and X_0 stands for an array of values of variables in $\pi(d)$. Given a policy S, we shall use $P_S(d|\pi(d))$ denote the conditional probability that represents the decision function $f_d \in S$ for d.

Given a policy for S for I, a decision node d of I can be viewed as a (deterministic) chance node with the conditional probability $P_S(d|\pi(d))$. The value nodes can always be viewed as chance nodes. Consequently, the influence diagram I itself can be viewed as a Bayesian net. Let us denote it by I_S. The prior joint probability (Zhang and Poole 1992) of the Bayesian net I_S is denoted by P_{I_S} or simply P_S. Explicitly, P_S is given by

$$P_S(C, D, U) = [\prod_{x\in C\cup U} P(x|\pi(x))][\prod_{x\in D} P_S(x|\pi(x))]. \quad (2)$$

For any $x \in C \cup U$, the conditional probability $P(x|\pi(x))$ is given as part of the specification of the influence diagram, while for any $x \in D$ the conditional probability $P_S(x|\pi(x))$ is given by the policy S.

Let v be a value node of I. Let P_S^v denote the marginal probability of v under P_S. The expectation $E_S(v)$[3] of v is given by

$$E_S(v) = \sum_{v_0\in\Omega_v} P_S^v(v_0)v_0.$$

The *value* $V_S(I)$ of I given a policy S is defined by

$$V_S(I) =_{def} \sum_{v\in U} E_S(v).$$

The *optimal expected value* $V(I)$ of I is defined by

$$V(I) =_{def} max\{V_S(I)|S \text{ is a policy for } I\}. \quad (3)$$

The optimal value of an influence diagram without value nodes is zero.

An *optimal policy* S_0 of I is one that satisfies

$$V_{S_0}(I) = V(I). \quad (4)$$

For an influence diagram without value nodes, all policies are optimal.

In this paper we shall only consider variables with finite frames. Hence there are only finite possible policies. Consequently, there always exists at least one optimal policy. *To evaluate an influence diagram* is to

1. find an optimal policy, and
2. find the optimal expected value.

4 STEPWISE-SOLVABILITY

The *decision function space* of a decision node d is the set $\{f : (\prod_{x\in\pi(d)} \Omega_x) \rightarrow \Omega_d\}$ of all the possible decision functions for d. The *policy space* of an influence

[3]Note that $E_S(v)$ is not a function of v. Rather it is a number. The notation is standard in Probability Theory, where the expectation of a random variable ξ is written as $E(\xi)$.

diagram is the Cartesian product of the decision function spaces of all the decision nodes.

A naive way to evaluate an influence diagram is to exhaustively search through the policy space. The complexity of doing so may be prohibitive. Hence, it would be desirable that an influence diagram be stepwise-solvable, i.e, it can be evaluated by searching through the decision function space for one decision node at a time, without ever having to consider the Cartesian product of the decision function spaces for any more than one decision nodes.

In this section, we shall formalize the concept of stepwise-solvability and illustrate that not all general influence diagrams are stepwise-decomposable. In the next section, we shall describe a subclass of influence diagrams that are stepwise-solvable.

To formally define the concept of stepwise-solvability, we need first introduce two other terms. Let d be a decision node in an influence diagram. Suppose, for a moment, that the decision functions for all other decision nodes are given. Each decision function for d, together with those decision functions for all the other decision nodes, constitutes a policy for the diagram, and thus results in an expected value of the diagram. The decision functions for d can be ranked according to the corresponding expected values. This ranking is called the *conditional ranking of the decision functions for d* given the decision functions for all other decision nodes.

To replace a decision node d by a *deterministic node* characterized by a function $f : \prod_{x \in \pi(d)} \Omega_x \rightarrow \Omega_d$ is to replace d by an new random node with the same name and the same frame, and to set $P(d|\pi(d))$ to be the conditional probability that represents f in the sense of equation (1).

An influence diagram is *stepwise-solvable* if it contains only one decision node, or there exists a decision node d, called a *candidate decision node*, such that

1. the conditional ranking of the decision functions for d given the decision functions for all other decision nodes is independent of those decision functions, and
2. if d is replaced by a deterministic node characterized by a decision function of d, the resulting influence diagram is stepwise-solvable.

If an influence diagram is stepwise-solvable, then it can be evaluated as follows. Let d be a candidate decision node. First find an optimal decision function for d in the way as specified in the next paragraph, replace d by a deterministic node characterized by the optimal decision function, resulting in another stepwise-solvable influence diagram I' with one less decision node. Then recursively apply the procedure to evaluate I'.

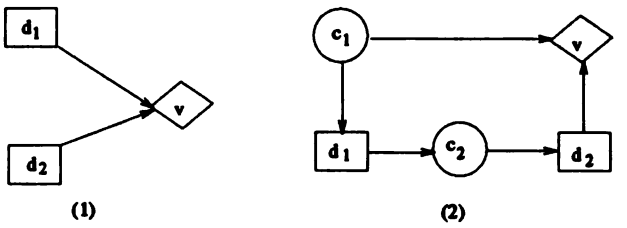


Figure 2: Two influence diagram skeletons that are not stepwise-solvable.

To find an optimal decision function for d, one first replaces all other decision nodes by deterministic nodes characterized by arbitrary decision functions, resulting in an influence diagram with only one decision node d. One can then find an optimal policy for d by searching through the decision function space of d. Because d is a candidate decision node, the optimal decision function for d such found does not depend on the choices of decision functions for the other decision nodes.

A directed graph $G = (V, A)$, with a partitioning of V into C, D, and U is called a *influence diagram skeleton*. From a given influence diagram skeleton, many influence diagrams can be defined by specifying the frames of the variables and the conditional probabilities of the variables in C and U given their parents. On the other hand, an influence diagram has only one corresponding influence diagram skeleton.

An influence diagram skeleton is *stepwise-solvable* if all the influence diagrams that can be defined from it are stepwise-solvable.

The two influence diagram skeletons shown in Figure 2 are not stepwise-solvable. One can easily construct from them influence diagrams that are not stepwise-solvable. The idea is that the value function can be so defined that it can not be separated into the summation or a multiplication of a function of d_1 and a function of d_2.

In the next section, we shall introduce the concept of stepwise-decomposability. One result of this paper is that stepwise-decomposability is equivalent to stepwise-solvability.

5 STEPWISE-DECOMPOSABLE INFLUENCE DIAGRAMS

In this section, we introduce stepwise-decomposable influence diagrams. We need the concepts of moral graph and m-separation. The *moral graph $m(G)$* of a directed graph G is the undirected graph obtained from G as follows: for each node of G, *marry* all the parents of d, i.e add an edge between every pair of its parents, and ignore all the directions. See Lauritzen and Speigehalter (1988) for more on this concept. If

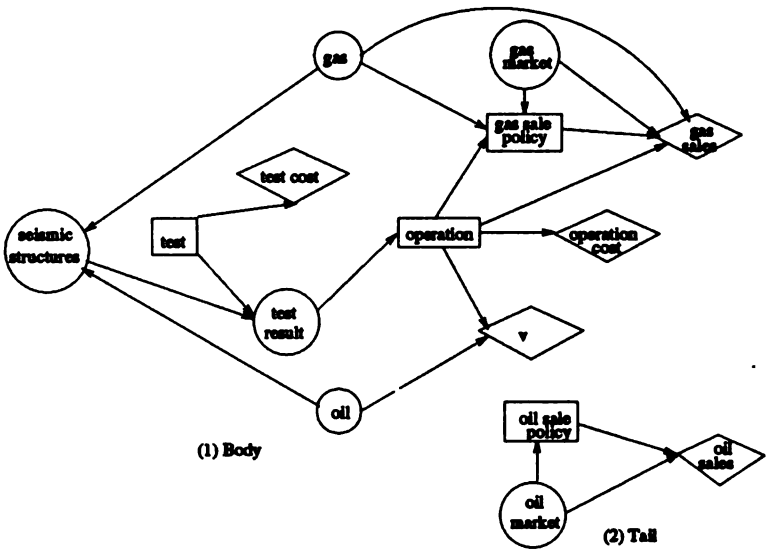


Figure 3: Tail and Body.

two nodes a and b of a directed graph G are separated in the moral graph $m(G)$ by a set X, we say that a and b are *m-separated* by X in G.

Let I be an influence diagram skeleton and d be a decision node in I. Let $nond(d)$ denote the set of decision nodes that are not descendents of d, and let $nond^*(d)$ be the set of the parents of the nodes in $nond(d)$ and the nodes in $nond(d)$ themselves.

An influence diagram skeleton is *stepwise-decomposable* if for each decision node d, the nodes in $nond^*(d)$ are either parents of d or are m-separated from d by the parents of d. An influence diagram is *stepwise-decomposable* if its skeleton is.

The influence diagram in Figure 1 is stepwise-decomposable.

Remark: A *no-forgetting influence diagram skeleton* (Shachter 1986) is an influence diagram skeleton in which (1) there is only one value node; (2) there is a directed path that contains all the decision nodes and the value node, with the value node coming last; and (3) each decision node and its parents are parents of all subsequent decision nodes. It is evident that the no-forgetting property implies stepwise-decomposability. So, all the theorems and algorithms for stepwise-decomposable influence diagrams apply to no-forgetting influence diagrams as well.

We now proceed to define the technical terms tail and body for stepwise-decomposable influence diagrams. Let $I = (V, A, \mathcal{P})$ be a stepwise-decomposable influence diagram, and let d be decision node. Let $m(G)$ be the moral graph of the directed graph $G = (V, A)$. Let G_1 be the undirected graph obtained from $m(G)$ by removing all the nodes in $\pi(d)$ of d. The *downstream Y_d of d (more accurately, of $\pi(d)$)* is the set of all the nodes of G_1 that are connected to d, with d itself excluded. The *upstream X_d of d* is the set of nodes of G_1 that are not connected to d.

Let G_2 be the undirected graph obtained from $m(G)$ by removing all the nodes in Y_d. Let $\pi'(d)$ be the set of nodes in $\pi(d)$ that are connected, in G_2, to at least one node in X_d.

The *tail* I_d^t of I w.r.t. d is the influence diagram obtained from I by restricting it to the set $Z =_{def} Y_d \cup \{d\} \cup (\pi(d) - \pi'(d))$. Particularly, the tail I_d^t contains and only contains the conditional probabilities of the chance an value nodes in Z.

The tail I_d^t of I is a parametric "influence diagram"[4] over Z. Some of the conditional probabilities of this influence diagram, hence its optimal expected value, contain potentially all the variables in $\pi'(d)$ as parameters. Let $V_d^t(\pi'(d))$ denote the optimal expected value of I_d^t.

The *body* I_d^b of I w.r.t. d is the influence diagram obtained from I by

1. restricting it to the set $X_d \cup \pi'(d)$. Particularly, the conditional probabilities of nodes not in $X_d \cup \pi'(d)$, or equivalently, of nodes in Z, are removed.
2. creating a new value node v and drawing an arc to v from each node in $\pi'(d)$, and

[4] Here, we put the term influence diagrams in quotes because the tail I_d^t may not be an influence diagram in the strict sense. The "prior probabilities" of some of the roots may not sum to 1. Fortunately, this fact does not affect our theory at all. For the sake of simplicity, we choose to stick to the term rather then come up a more accurate one.

3. setting the conditional probability $P(v|\pi'(d))$ to be the conditional probability that represents the function $V_d^t(\pi'(d))$ in the sense of equation (1).

Figure 3 illustrate the concepts of tail and body. In the influence diagram shown in Figure 1, "oil-sale-policy" (osp) is a decision node. The tail w.r.t. "oil-sale-policy" is shown in Figure3 (2), which contains the prior probability $P(oil\text{-}market)$ and the conditional probability $P(oil\text{-}sales|operation, oil\text{-}sale\text{-}policy, oil\text{-}market, oil)$, where "operations" and "oil" are parameters. Let $V_{osp}^t(operation, oil)$ be the optimal expected value of the tail. The body w.r.t. "oil-sale-policy" is shown in Figure 3 (1), where the conditional probability $P(v|oil, operation)$ of the newly introduced value node v given "oil" and "operation" represents the function $V_{osp}^t(operation, oil)$.

Note that if an influence diagram contain only one decision node, then its body does not contain any decision node. This fact will be used in the evaluation algorithm of the next section.

A influence diagram is *simple* if it contains only one decision node. The influence diagram in Figure 3 (2) is simple.

A decision node is a *leaf decision node* if none of it descendants are decision nodes. The following proposition is straightforward.

Proposition 1 *Suppose an influence diagram I is stepwise-decomposable and d is a leaf decision node of I. Then*

1. *The tail I_d^t of I w.r.t. d is a simple influence diagram.*
2. *The body I_d^b of I w.r.t. d is a stepwise-decomposable influence diagram.*

6 EVALUATING STEPWISE-DECOMPOSABLE INFLUENCE DIAGRAMS

In this section, we present a recursive algorithm which evaluates stepwise-decomposable influence diagrams in a stepwise fashion by considering one decision at a time and by computing conditional probabilities in local Bayesian nets that correspond to sections of the diagram. Let us begin with simple influence diagrams.

6.1 EVALUATING SIMPLE INFLUENCE DIAGRAMS

Suppose I is a simple influence diagram. Let d denote the only decision node and let $v_1, ..., v_m$ be the value nodes. An optimal decision function for d can be found by searching through the entire decision function space for d. More specifically, we can proceed as follows.

For each possible decision function $f : \prod_{x\in\pi(d)} \Omega_x \rightarrow \Omega_d$ for d, replace d by a deterministic node characterized by f, resulting in a Bayesian net I_f. For i from 1 to m, compute the marginal probability $P_{I_f}(v_i)$ of v_i in the Bayesian net I_f, and the expectation $E_{I_f}(v_i)$ from $P_{I_f}(v_i)$. The value $V_f(I)$ of I given the decision function f can now be obtained by using

$$V_f(I) = \sum_{i=1}^{m} E_{I_f}(v_i).$$

After the value $V_f(I)$ having been computed for all the decision functions f for d, pick one such function f_0 whose corresponding value $V_{f_0}(I)$ is the maximum. Then f_0 is an optimal decision function for d.

The above procedure has transformed the problem of evaluating the simple influence diagram I into the task of computing the marginal probabilities in the Bayesian nets I_f. The problem of computing marginal probabilities in Bayesian nets has been under extensive study in recent years. See Pearl (1986), Lauritzen and Speigehalter (1988), Shafer and Shenoy (1988), Jensen *et al* (1990), and Zhang and Poole (1992) for elegant algorithms.

Note that as a by-product of finding an optimal policy for a simple influence diagrams, its optimal expected value is also determined.

6.2 THE GENERAL CASE

Let I be a stepwise-decomposable influence diagram and let d be a leaf decision of I. The tail I_d^t of I is a simple influence diagram[5]. The parents of d in I_d^t are the nodes in $\pi(d) - \pi'(d)$. Some conditional probabilities of I_d^t may possibly contain nodes in $\pi'(d)$ as parameters. Thus, an optimal decision function for d in the tail I_d^t depends on the nodes in $\pi'(d)$, as well as nodes in $\pi(d) - \pi'(d)$. For each value of $\pi'(d)$, we get an optimal decision function for d in I_d^t. Let $\pi'(d)$ run through all its possible values, we get a function from $\prod_{x\in\pi(d)} \Omega_x$ to Ω_d. Let us call the function an *optimal decision function* for d in the original influence diagram I.

Two influence diagrams are *value-equivalent* if they have the same optimal expected value.

Theorem 1 *Suppose I is a stepwise-decomposable Influence diagram. Then*

1. *An optimal decision function for a leaf decision node d and an optimal policy of the body I_d^b of I w.r.t. d constitute an optimal policy for I itself.*

[5]See footnote 4.

2. *The influence diagrams I_d^b and I are value-equivalent.*

The proof of the theorem can be found in the appendix.

The theorem suggests the following procedure EVALUATE(I) for evaluating a stepwise-decomposable influence diagram I.

Procedure EVALUATE(I):

- Input: I – a stepwise-decomposable influence diagram.
- Ouput: an optimal policy for I and the optimal expected value $V(I)$ of I.

1. If I does not contain decision nodes, then N-EVALUATE(I),
2. Else find a leaf decision node d, and
   - S-EVALUATE(I_d^t),
   - EVALUATE(I_d^b),
   - $V(I) \leftarrow V(I_d^t)$.

where S-EVALUATE is a procedure that evaluates simple influence diagrams. It takes a simple influence diagram, returns its optimal expected value and an optimal policy. N-EVALUATE is a procedure that computes the expected value of an influence diagram without decision nodes.

Several remarks are in order.

1. Theorem 1 and Proposition 1 together have shown that stepwise-decomposable influence diagrams are stepwise-solvable. So, an influence diagram skeleton is stepwise-solvable if it is stepwise-decomposable. For any influence diagram skeleton that is not stepwise-decomposable, it is not hard to build from it an influence diagram that is not stepwise-solvable[6]. Therefore, as far as influence diagram skeletons are concerned, stepwise-decomposability and stepwise-solvability are equivalent.
2. All the tail influence diagrams encountered by the EVALUATE procedure are simple, and they correspond to disjoint sections of the original influence diagram. Furthermore evaluating a simple influence diagram is just a matter of computing marginal probabilities in Bayesian nets[7]. It is in this sense that we say we have transformed the problem of evaluating stepwise-decomposable influence diagrams into problems of computing marginal probabilities in local Bayesian nets.
3. In the context of no-forgetting influence diagrams, Shachter (1988, 1990) has pointed out an optimal decision for a leaf decision d can be computed from $E(v|d, \pi(d))$. After an optimal decision function d is computed, he suggests to replace d with a deterministic node characterized by the decision function, and repeat the procedure for the remaining decision nodes. This approach is inefficient, because computations in the tails are repeated, possibly many times.
4. In implementing the procedure EVALUATE(I), one can use a number of approaches for computing marginal probabilities in Bayesian nets. On the other hand, the node reduction algorithms in the literature (Shachter 1986, Smith 1989, Shenoy 1990, and Ndilikilikesha 1991) correspond to only one of those approaches, and not necessarily the most efficient one.

7 REMOVABLE DECISION DEPENDENCY ARCS

When disussing the evaluation of no-forgetting influence diagrams, Shachter (1988, 1990) and Tatman and Shachter (1990) have pointed out that an optimal decision function for a decision node d does not necessarily depend on all the parents of d. Shachter (1988, 1990) has developed a linear time algorithm to find, for a given decision node d, parents of d that would not appear in an optimal decision function for d.

In this section, a formal definition of the concept of removable decision dependency arcs are provided. One type of removable decision dependency arcs, namely unaccompanied arcs, will be graphically characterized independent of any evaluation algorithm. This characterization leads to a simple algorithm for finding all the unaccompanied decision dependency arcs. It can be proved that unaccompanied arcs are the only removable decision dependency arcs that can be recognized using only graphical information.

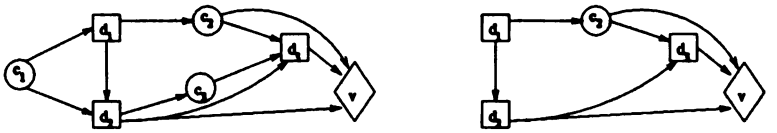
In our terminology, Shachter's algorithm only finds the unaccompanied arcs for a given decision node, while ours finds all the unaccompanied arcs in one sweep.

Let us first introduce the concept of decision equivalence for influence diagrams. Two influence diagrams with the same nodes are *decision-equivalent* if they are value-equivalent and the parents of value nodes and chance nodes are the same in both influence diagrams.

Two influence diagram skeletons are *decision-equivalent* if influence diagrams built from them by specifying the same frames for the same nodes and

[6] If an influence diagram skeleton is not stepwise-decomposable, then there are portions of it looks like one of the diagrams in Figure 2. Thus, influence diagrams that are not stepwise-decomposable can be constructed from the skeleton.

[7] As stated in footnote 4, the tail I_d^t may not be an influence diagram in the strict sense. So, the "Bayesian nets" here may not be Bayesian nets in the strict sense either. The "prior probabilities" of some of the roots may not sum to 1. Fortunately, this fact has little impact on the computations involved.



(1) (2)

Figure 4: Removable arcs: The arc $c_3 \rightarrow d_3$, $c_1 \rightarrow d_2$ and $c_1 \rightarrow d_1$ are removable.

the some conditional probabilities for the same value nodes and chance nodes are decision-equivalent.

Intuitively, two influence diagrams being decision-equivalent means that the influence diagrams are identical, except that the decisions may be made on the basis of potentially different information. The conditional probabilities are the same, the decisions are to be made are the same, and if optimal decisions are made in both cases, the resulting expected values are the same.

Suppose I is a stepwise-decomposable influence diagram. A decision dependence arc $a \rightarrow d$ from a node a to a decision node d is *removable* if the removal of the arc for the influence diagram results in a stepwise-decomposable influence diagram that is decision-equivalent to I.

Let a be a parent of a decision node d in a stepwise-decomposable influence diagram I. Arcs from a to nodes in Y_d and arcs from nodes in Y_d to a are said to *accompany* the arc $a \rightarrow d$. The arc $a \rightarrow d$ can be accompanied by any number of arcs. When the is accompanied by no arcs, we say that it is *unaccompanied*.

In the influence diagram shown in Figure 4, $Y_{d_3} = \{v\}$. The arc $c_2 \rightarrow d_3$ is accompanied by the arc $c_2 \rightarrow v$. There is no arc either from c_3 to v or from v to c_3. So, $c_3 \rightarrow d_3$ is unaccompanied.

Theorem 2 *Suppose I is a stepwise-decomposable influence diagram. Let I' be the influence diagram obtained from I by removing some unaccompanied decision dependency arcs. Then I' is stepwise-decomposable and it is decision-equivalent to I. In other words, unaccompanied arcs are removable.*

Proof: It suffices to show that I and I' are value-equivalent. We can assume, without the loss of generality, that there has been only one unaccompanied arc removed from I. Because of Theorem 1, we can assume that the decision dependency arc $a \rightarrow d$ removed is such that d is a leaf decision node.

Because $a \rightarrow d$ is unaccompanied, none of the conditional probabilities of the tail I_d^i contains the node a as a parameter. Thus there exists an optimal policy on d that does not depend on a. Therefore there is an optimal policy for I, in which the decision function for d does not depend on a. Let us call such a policy an *$a \rightarrow d$ ignoring policy*.

Any policy of I' can be thought as a policy for I, an $a \rightarrow d$ ignoring policy. So, the optimal expected value of I' is less than or equal to that of I. On the other hand, an $a \rightarrow d$ ignoring policy for I can be viewed as a policy for I' as well, which yields the same expected value for I' as for I. So, the optimal expected value of I can not be strictly greater than that of I'. Therefore I and I' have the same optimal expected value. The theorem is thus proved. □

In the influence diagram shown in Figure 4 (1), $c_3 \rightarrow d_3$ is unaccompanied, and hence is removable. The arc $c_1 \rightarrow d_1$ is not unaccompanied (removable) at the beginning. But it becomes unaccompanied (removable) after the removal of $c_1 \rightarrow d_2$, which is unaccompanied (removable). We call such decision dependency arc potentially *unaccompanied arcs*.

Here is an algorithm which finds all unaccompanied and potentially unaccompanied decision dependency arcs of a stepwise-decomposable influence diagram skeleton I. The second argument *Collector* is initially empty.

Procedure UNACCOMPANIED-ARCS(I, *Collector*)

1. **If** there is no decision node in I, return *Collector*.
2. **Else** find a leaf decision node d, compute Y_d, and
   - Find all the arcs from $\pi(d)$ that are unaccompanied. Let A be the set of all such arcs, and let B be the set of the nodes in $\pi(d)$ that appear in the arcs of A.
   - Set *Collector*1 $= A \cup$ *Collector*.
   - Remove from I all the nodes in Y_d; introduce an new node v into the resulting diagram; and draw any arc from every node in $\pi(d) - B$ to v. Let I' be the resulting skeleton.
   - UNACCOMPANIED(I', *Collector*1).

Going back to our example, after the removal of $c_3 \rightarrow d_3$, the the node c_3 becomes barren, and hence can be removed (Shachter 1986). After the removal of unaccompanied arcs and the barren node created because

of the removals, the influence diagram in Figure 4 (1) becomes the one in Figure 4 (2).

Remark. For any decision dependency arc in a stepwise-decomposable influence diagram skeleton, it is not hard to build an influence diagram from the skeleton such that the arc is not removable. Take the skeleton in Figure 4 (2) for instance, the arc $c_2 \rightarrow d$ is accompanied by $c_2 \rightarrow v$. We can define the conditional probability $P(v|c_2, d_3, d_2)$ in such way that optimal polices for d_3 always depend on c_2. Thus, unaccompanied arcs are the only removable arcs that can be recognized using only graphical information.

8 CONCLUSIONS

Stepwise-decomposable influence diagrams have been introduced. Modeling with stepwise-decomposable influence diagrams enables use to represent knowledge about a piece of information being irrelevant to a decision, and consequently enables us to prevent the decision table from being unnecessarily large at the problem formulation stage. Evaluating stepwise-decomposable influence diagrams amounts to computing conditional probabilities in local Bayesian nets that correspond to disjoint sections of the original influence diagrams. There are a number of ways to compute conditional probabilities in Bayesian net. Shachter's arc reversal and node reduction algorithm corresponds to only one of those ways, and not necessarily the most efficient one. Stepwise-decomposable influence diagrams also prove to be a convenient setting for studying removable decision dependency arcs. Consequently they allow us to prevent, in the evaluation process, the decision tables from being unnecessarily large.

Stepwise-decomposability is an interesting property. It captures the essence of the concept of stepwise-solvability from the viewpoint of graph theory. It leads to a simple and insightful mathematical theory of influence diagrams, particularly efficient and easy to understand algorithms. It also leads to a natural way of exploiting asymmetries in evaluating influence diagrams (Qi and Poole 1992).

Acknowledgements

The first author is grateful to Glenn Shafer and Prakash Shenoy who first brought the influence diagram literature to my attention. Discussions with and comments from Brent Boerlage, Craig Boutilier, Mike Horsch, Keiji Kanazawa, Runping Qi and Gregory Provan have been valuable. Comments from reviewers of KR'92 have also been useful. Research is supported by NSERC Grant OGPOO44121.

Appendix: Proof of Theorem 1

To prove Theorem 1, we need the following lemma of stepwise optimization.

Let X, Y, and Z be three mutually disjoint sets of variables with finite frames. Let $\mathcal{F}$ be a finite collection of real-valued non-negative functions of X and Y, and $\mathcal{G}$ a finite collection of real-valued non-negative functions of Y and Z. Let $h(X,Y)$ be a non-negative function of X and Y.

The following proposition says that to find two functions f and g to maximize the sum $\sum_{X,Y} f(X,Y)(\sum_Z g(Y,Z) + h(X,Y))$, one can proceed by trying first to find a function g_0 that maximizes the sum $\sum_Z g(Y,Z)$ for every value of Y, and then to find a function f that maximizes the sum $\sum_{X,Y} f(X,Y)\{\sum_Z g_0(Y,Z) + h(X,Y)\}$.

Lemma 1 *If $g_0 \in \mathcal{G}$ is such that for any value of Y,*

$$\sum_Z g_0(Y,Z) = max_{g \in \mathcal{G}} \sum_Z g(Y,Z),$$

and if $f_0 \in \mathcal{F}$ is such that

$$\sum_{X,Y} f_0(X,Y)\{\sum_Z g_0(Y,Z) + h(X,Y)\} = max_{f \in \mathcal{F}} \sum_{X,Y} f(X,Y)\{\sum_Z g_0(Y,Z) + h(X,Y)\},$$

then

$$\sum_{X,Y,Z} f_0(X,Y)(\sum_Z g_0(Y,Z) + h(X,Y)) = max_{f \in \mathcal{F}, g \in \mathcal{G}} \sum_{X,Y} f(X)(\sum_Z g(Y,Z) + h(X,Y).$$

Proof: It is evident that the left hand side is less than or equal to the right hand side. So, we need only show that the right hand side is no larger than the left hand side. Actually,

$$\begin{aligned} & max_{f \in \mathcal{F}, g \in \mathcal{G}} \sum_{X,Y} f(X,Y)(\sum_Z g(Y,Z) + h(X,Y)) \\ & \leq max_{f \in \mathcal{F}, g \in \mathcal{G}}\{\sum_{X,Y} f(X,Y)(\sum_Z g_0(Y,Z) + h(X,Y))\} \\ & = max_{f \in \mathcal{F}} \sum_{X,Y} f(X,Y)(\sum_Z g_0(Y,Z) + h(X,Y) \\ & = \sum_{X,Y} f_0(X,Y)(\sum_Z g_0(Y,Z) + h(X,Y)). \end{aligned}$$

Therefore, the lemma is proved. □

Suppose $I = (V, A, \mathcal{P})$ is a stepwise-decomposable influence diagram. Let d be leaf decision node of I. Let sets X_d, Y_d and $\pi'(d)$ be as defined in section 5. Let $X = X_d$, $Y = \pi'(d)$, and $Z = Y_d \cup \{d\} \cup (\pi(d) - \pi'(d))$. Then X, Y and Z constitute a partition of the set V of all the nodes of I.

Lemma 2 *No node in $\pi'(d)$ is a parent to any node in X_d.*

Proof: Any node $a \in \pi'(d)$ is a parent of d. If a were also a parent of some $x \in X_d$, then x and d would be connected by an edge in the moral graph $m(G)$, where $G = (V, A)$ is the directed graph underlying I. This Contradicts the definition of X_d. □

Let $v_1, ..., v_n$ be all the value nodes of I, among them $v_1, ..., v_m$ $(m \leq n)$ are all the values nodes in Z (i.e in the tail I_d^t), and $v_{m+1}, ..., v_n$ are in $X \cup Y$. Let v be the newly introduced value node of the body I_d^b.

Define functions $h(X, Y) =_{def} v_{m+1} + ... + v_n$, and

$$f_S(X, Y) =_{def} \prod_{x \in ((C \cup U) \cap (X \cup Y))} P(x|\pi(x)) \prod_{x \in (D \cap (X \cup Y))} P_S(x|\pi(x)),$$

$$g_S(Y, Z) =_{def} \prod_{x \in ((C \cup U) \cap Z)} P(x|\pi(x)) P_S(d|\pi(d)) \sum_{i=1}^{m} v_i.$$

Define the *prebody* I_d^p of I w.r.t. d to be the influence diagram obtained from I by restricting it to $X \cup Y$, i.e by removing all the node outside $X \cup Y$ and their conditional probabilities when applicable. Given a policy S, let f_d be the decision function for d, and let $S_d^p = S - \{f_d\}$. Then S_d^p is a policy for the prebody I_d^p.

The joint probability P_S that S determines over all the nodes V of I will be rewritten as $P_{S,I}$ from now on. The meanings of the symbols $E_{S,I}$, $P_{S_d^p, I_d^p}$, and $E_{S_d^p, I_d^p}$ are defined accordingly.

Because of the definition of a prebody, we have that for any value node v_j $(m < j \leq n)$ in the prebody I_d^p,

$$E_{S_d^p, I_d^p}(v_j) = \sum_{X,Y} f_S(X, Y) v_j. \tag{5}$$

Lemma 3 *1. For any node $x \in X$ (i.e in the prebody I_d^p),*

$$P_{S,I}(x) = P_{S_d^p, I_d^p}(x).$$

2. For any value node $v_j \in X$,

$$E_{S,I}(v_j) = E_{S_d^p, I_d^p}(v_j).$$

Proof: Because of Lemma 2, the first assertion can be obtained by a series of barren node removal (see, for example, Shachter 1988). The second assertion follows from the first one immediately. □

This corollary follows from Lemma 3 and equation (5).

Corollary 1 *For any value node $v_j \in X$,*

$$E_{S,I}(v_j) = \sum_{X,Y} f_S(X, Y) v_j.$$

Amongst all the decision functions in S, $g_S(Y, Z)$ only depends on the decision function f_d for d. On the other hand, $f_S(X, Y)$ depends on the decision functions in S_d^p only. In the sequel, we shall rewrite f_S as $f_{S_d^p}$, and g_S as g_{f_d}.

Lemma 4 *Let $V_d^t(Y)$ (remember that $Y = \pi'(d)$) be as defined in section 5. Then*

$$max_{f_d} \sum_{Z} g_{f_d}(Y, Z) = V_d^t(Y).$$

Proof: The proof readily follows from the definition of the function g_S or g_{f_d}. □

Proof of Theorem 1: Because of the definition of the functions $f_{S_d^p}$ and g_{f_d} and Corollary 1, we have

$$\begin{aligned}
\sum_{i=1}^{n} E_{S,I}(v_i) &= \sum_{i=1}^{m} E_{S,I}(v_i) + \sum_{j=m+1}^{n} E_{S,I}(v_i) \\
&= \sum_{X,Y,Z} f_{S_d^p}(X, Y) g_{f_d}(Y, Z) + \\
&\quad \sum_{X,Y} \sum_{j=m+1}^{n} f_{S_d^p}(X, Y) v_i \\
&= \sum_{X,Y} f_{S_d^p}(X, Y) (\sum_{Z} g_{f_d}(Y, Z) + h(X, Y))
\end{aligned}$$

Therefore the theorem follows from Lemma 1 and lemma 4.

References:

R. A. Howard, and J. E. Matheson (1984), Influence Diagrams, in *The principles and Applications of Decision Analysis*, Vol. II, R. A. Howard and J. E. Matheson (eds.). Strategic Decisions Group, Menlo Park, California, USA.

F. V. Jensen, K. G. Olesen, and K. Anderson (1990), An algebra of Bayesian belief universes for knowledge-based systems, *Networks*, 20, pp. 637 - 659.

S. L. Lauritzen and D. J. Speigehalter (1988), Local computations with probabilities on graphical structures and their applications to expert systems, *Journal of Royal Statistical Society B*, 50: 2, pp. 157 - 224.

S. L. Lauritzen, A. P. Dawid, B. N. Larsen, and H. G. Leimer (1990), Independence properties of directed Markov fields, *Networks*, Vol. 20, pp. 491-506.

A. C. Miller, M. W. Merkofer, R. A. Howard, J. E. Matheson, and T. R. Rice (1976), Development of automated aids for decision analysis, Meno Park, CA: Stanford Research Institute.

J. Pearl (1988), *Probabilistic Reasoning in Intelligence Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, Los Altos, CA.

R. Qi and D. Poole (1992), *Exploiting asymmetries in influence diagram evaluation*, in preparation.

P. Ndilikilikesha (1991), Potential Influence Diagrams, Working Paper No. 235, Business School, University of Kansas.

H. Raiffa, (1968), *Decision Analysis*, Addison-Wesley, Reading, Mass.

G. Shafer and P. Shenoy (1988), Local computation in hypertrees, Working Paper No. 201, Business School, University of Kansas.

R. Shachter (1986), Evaluating Influence Diagrams, *Operations Research*, 34, pp. 871-882.

R. Shachter (1988), Probabilistic Inference and Influence Diagrams, *Operations Research*, 36, pp. 589-605.

R. Shachter (1990), An ordered Examination of Influence Diagrams, *Networks*, 20, pp. 535-563.

P. P. Shenoy, (1990), Valuation-Based Systems for Bayesian Decision Analysis, Working Paper No. 220, Business School, University of Kansas.

J. Q. Smith (1989), Influence Diagrams for Statistical Modeling, *Ann. Stat.*, 17, pp. 654-672.

J. A. Tatman and R. Shachter (1990), Dynamic programming and influence diagrams, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, pp. 265-279.

L. Zhang and D. Poole (1992), Sidestepping the triangulation problem in Bayesian net computations, in *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, July 17-19, San Jose.

A Logic for Approximate Reasoning

Daphne Koller
Stanford University
Stanford, CA 94305
email: daphne@cs.stanford.edu

Joseph Y. Halpern
IBM Almaden Research Center
San Jose, CA 95120
mail: halpern@almaden.ibm.com

Abstract

We investigate the problem of reasoning with imprecise quantitative information. We give formal semantics to a notion of approximate observations, and define two types of entailment for a knowledge base with imprecise information: a cautious notion, which allows only completely justified conclusions, and a bold one, which allows jumping to conclusions. Both versions of the entailment relation are shown to be decidable. We investigate the behavior of the two alternatives on various examples, and show that the answers obtained are intuitively desirable. The behavior of these two entailment relations is completely characterized for a certain sublanguage, in terms of the logic of true equality. We demonstrate various properties of the full logic, and show how it applies to many situations of interest.

1 Introduction

In almost any situation involving quantitative information, some of the information is bound to be approximate and imprecise. Moreover, such imprecision can easily cause inconsistencies. Consider, for example, the following knowledge base KB:

> Bill is 1.8 meters tall.
> John is half a head taller than Bill.
> A head is 0.2 meters.

Although the information in this knowledge base is not intended to be completely precise, we might nevertheless want to conclude from it that John is 1.9 meters tall. It is clear that we want to view this conclusion as being only an approximation of the truth. In particular, if we later obtain the additional piece of information "John is 1.88 meters tall," we do not want to conclude that the resulting knowledge base KB' is inconsistent; rather, we view this as a problem due to inaccurate measurement. This shows that we cannot interpret "is" in approximate observations as true numeric equality, because we would end up deducing that the above knowledge base is inconsistent, thus enabling arbitrary conclusions.

The need for dealing with approximate information arises in many other contexts. We often want to say that a certain quantity (such as a probability) is very close to zero, without committing to a particular value. The technique of ϵ-semantics [Pea88] is based on this concept (see Section 6.3). When dealing with statistical information, we often use statements of the form "90% of birds fly;" however, we do not wish to infer that the number of birds is divisible by 10, as we could if we interpret this statement as "*precisely* 90% of birds fly." It is more appropriate to interpret it as "*approximately* 90% of birds fly." (See [GHK92] for a thorough discussion.) Problems relating to the intransitivity of the perceptual indistinguishability relation in human observations [SKLT89] can also be formulated and circumvented using approximate equality (see Section 6.2).

In this paper, we introduce a logic which enables us to deal with and reason about imprecise information and the inconsistencies that usually accompany it. Our logic extends standard real arithmetic with notions of approximate equality and inequality. We formalize approximate equality to allow some small but unspecified discrepancy between the values being compared.

Our main interest is in making deductions from knowledge bases, so we focus here on what we call *approximate entailment*, where we view "KB approximately entails φ" as meaning that we have reasonable justification for concluding φ given the knowledge base KB. For example, if we are interested in buying John a jacket, and we are given the knowledge base KB above, we would certainly think it justified to proceed under the assumption that John is about 1.9 meters tall. The problem becomes more difficult when we ask for inferences from the extended knowledge base KB' above.

We present two different entailment relations that we call *cautious entailment* and *bold entailment*. They differ in the degree to which they allow the agent to "leap to conclusions;" i.e., in the degree of default reasoning they incorporate. The knowledge base KB' cautiously entails that "John is approximately between 1.88 and 1.9 meters tall." Thus, given contradictory information, the cautious approach assumes the answer could be anywhere in between. On the other hand, the bold approach, given the same knowledge base, would be able to conclude that "John is approximately h meters tall" for each h between 1.88 and 1.9; any reasonable number can be used as a "guesstimate."

From this example, it is clear that cautious entailment is nonmonotonic: by adding additional information to the knowledge base KB, we lose the ability to deduce that "John is approximately 1.9 meters tall." On the other hand, bold entailment is usually monotonic in the sense that adding new data to the knowledge base does not force us to withdraw conclusions. From the knowledge base KB' we can still deduce that "John is approximately 1.9 meters tall." However, the bold logic is not a standard monotonic logic. Although we can deduce both that "John is approximately 1.9 meters tall" and that "John is approximately 1.88 meters tall," we cannot deduce their conjunction (see Section 5.2 for more details). At first, this might seem strange. But the intuition here is that, although we can work with any reasonable assumption about John's height, we do not want to work with contradictory assumptions simultaneously.

Both types of entailment can be reduced to the validity of a formula in the language of *real closed fields* [Tar51], and therefore are decidable. The decision procedure, however, does not give us much insight into the properties of entailment. To gain this insight, we consider several examples, and present general properties of our notion of approximate entailment. These show that approximate entailment agrees with our intuition in many situations. For example, we show that inferences made by either one of our entailment relations are always consistent with those obtained by taking approximate equality to be true equality. However, if the knowledge base is inconsistent with equality, as with KB' above, it entails only "reasonable" conclusions. We provide an elegant characterization of these entailment relations for a large sublanguage of our full language; in particular, the characterization justifies our choice of the names "bold" and "cautious." As a corollary to this characterization, we show that if our data is consistent even if approximate equality is treated as true equality, then we typically get *precisely* the conclusions that we get from true equality. Our characterization also shows that, for a large subclass of formulas, cautious entailment reduces to a variant of *preference semantics* [Sho87] (see Section 5.2).

While most of the paper focuses on issues concerning measurement, our approach is actually much more general. Given a notion of exact inference from a knowledge base with precise information, we can use our framework to extend it to a notion of approximate inference from a knowledge base of imprecise information. The notion of exact inference could well be probabilistic or nonmonotonic. In particular, we can apply these ideas to ϵ-semantics ([Pea88, GMP90]) and to the problem of computing degrees of belief from statistical information [GHK92] (see Section 6.3).

2 Syntax and Semantics

Since we want to focus on the basic issues arising from the problem of approximate numerical information, we restrict ourselves to considering a relatively simple framework where these issues arise. We begin with a core language $\mathcal{L}$, consisting of:

- the standard arithmetic operations of $+, -, \times, /$,
- the standard equality and inequality relations $=$ and $\leq$,
- a constant symbol d_r for each real number r (in our examples, we typically write, say, 0.1, rather than $d_{0.1}$),
- a countable collection $c_1, c_2, \ldots$ of uninterpreted constant symbols.

We form the set of *terms* by closing off the constants under $+, -, \times$ and $/$. The set $\mathcal{E}$ of *precise expressions* consists of formulas of the form $t = t'$ and $t \leq t'$ for terms t and t'. The language $\mathcal{L}$ is formed by closing off $\mathcal{E}$ under conjunction, disjunction, and negation.

In order to form the *approximate language* $\mathcal{L}^{\approx}$, we augment the language $\mathcal{L}$ with the *approximate equality and inequality* relations $\approx$ and $\preceq$. The set $\mathcal{A}$ of *approximate expressions* consists of formulas of the form $t \approx t'$ and $t \preceq t'$ for terms t and t'. The language $\mathcal{L}^{\approx}$ is formed by closing off $\mathcal{E} \cup \mathcal{A}$ under conjunction, disjunction, and negation.

We interpret $\mathcal{L}$ in the standard fashion. Terms are interpreted over the reals, with an additional undefined value ι (used to deal with the problem of division by zero). The symbols $+, -, \times, /, =, \leq$ receive their standard interpretation (extended to deal with ι), and the constant d_r is interpreted as the real number r.

Definition 2.1: A model v for $\mathcal{L}$ is a function assigning an element in $\mathbf{R} \cup \{\iota\}$ to each term, and a truth value to each formula, as follows:

- for each uninterpreted constant c_i, $v(c_i) \in \mathbf{R}$,
- for each constant d_r, $v(d_r) = r \in \mathbf{R}$,
- for each term $t \circ t'$, where $\circ \in \{+, -, \times, /\}$, we have $v(t \circ t') = v(t) \circ v(t')$ in the standard way, with the following exceptions:

- if $v(t') = 0$, then $v(t/t') = \iota$,
- if $v(t) = \iota$ or $v(t') = \iota$, then $v(t \circ t') = \iota$.

- if $\sim$ is one of $=$ or $\leq$, then $v(t \sim t')$ is *true* if both $v(t)$ and $v(t')$ are in $\mathbb{R}$ and $v(t) \sim v(t')$; otherwise $v(t \sim t')$ is *false*,
- v is extended to Boolean combinations of precise expressions in the standard way. ∎

One might wonder why we take division to be a primitive operation in our language (at the cost of having to deal with ι), rather than defining it in terms of multiplication. The problem arises due to subtle interactions between division and the semantics of approximate equality, in such terms as a/b where $b \approx 0$. This is particularly relevant in such applications as ϵ-semantics. It turns out that the only way to handle such expressions appropriately is to allow division as a primitive operation (see Section 6.3).

To understand the interpretation of $\approx$ and $\preceq$, we first need to consider how we want to interpret a statement such as: "Bill is approximately 1.8 meters tall." We view such a statement as describing a measurement of Bill's height, taken with some unknown degree of inaccuracy. Thus, we take this to mean that Bill's height is within τ of 1.8 meters, for some unknown tolerance τ. In general, the tolerances for different measurements may be completely independent. We enforce this in our semantics by interpreting $\approx$ and $\preceq$ in a nonstandard, context-dependent manner; for each expression $e \in \mathcal{A}$, there is a tolerance $\tau(e)$ associated with e.[1]

Definition 2.2: The *tolerance function* τ is a function from $\mathcal{A}$ to $\mathbb{R}^+ = [0, \infty)$. Let $\mathcal{T}$ denote the set of tolerance functions. ∎

In this definition, we have chosen to allow 0 as a legal tolerance value; that is, the range of a tolerance function is $[0, \infty)$, not $(0, \infty)$. While this issue may seem minor, it has a number of side effects. For one thing, it allows us to state stronger theorems, with simpler proofs. But it also affects our ability to make certain inferences. We discuss this issue further in Section 5.1.

In order to relate the meaning of expressions in $\mathcal{L}^{\approx}$ to the semantics of $\mathcal{L}$, we need the following definition.

Definition 2.3: For a formula $\varphi \in \mathcal{L}^{\approx}$, and a fixed tolerance function τ, we define $\varphi[\tau] \in \mathcal{L}$ to be the same as φ, except that every approximate expression $t \approx t'$ is replaced by the expression $|t - t'| \leq d_{\tau(t \approx t')}$ and each expression $t \preceq t'$ is replaced by $(t - t') \leq d_{\tau(t \preceq t')}$. ∎

For example, if φ is $c_1 \approx c_2$, and τ is such that $\tau(c_1 \approx c_2) = 0.1$, then $\varphi[\tau] = |c_1 - c_2| \leq d_{0.1}$. Let τ_0 to be the tolerance function that assigns tolerance 0 to all expressions. Note that $\varphi[\tau_0]$ is precisely the result of interpreting all occurrences of "approximately equals" in φ as true equality. We say that φ is *consistent with equality* if $\varphi[\tau_0]$ is consistent.

Definition 2.4: An *augmented model* M for $\mathcal{L}^{\approx}$ is a pair (v, τ), where v is a model for $\mathcal{L}$ and τ is a tolerance function. For a formula $\varphi \in \mathcal{L}^{\approx}$, we define $M \models \varphi$ if $v \models \varphi[\tau]$. ∎

Note that $\tau(e)$ for expressions e that do not appear in a formula φ has no affect on the truth value of φ: if τ and τ' agree on all expressions that appear in φ, then for any v, we have $(v, \tau) \models \varphi$ iff $(v, \tau') \models \varphi$.

We define validity for $\mathcal{L}^{\approx}$ as usual: ψ is valid if $M \models \psi$ for all models M. The validity problem for $\mathcal{L}^{\approx}$ is of little interest, as the following example suggests.

Example 2.5: Let φ be $(c \approx 1) \Rightarrow (2 \times c \approx 2)$, and the model $M = (v, \tau)$, for $v(c) = 1.1, \tau(c \approx 1) = 0.1$, and $\tau(2 \times c \approx 2) = 0.15$. Then $M \not\models \varphi$, and therefore φ is not valid. ∎

In fact, as the following theorem shows, if we restrict attention to (Boolean combinations of) approximate expressions not involving division (which allows us to avoid all the complications of dealing with ι), the only valid formulas are those that are propositionally valid if we treat every approximate expression as a distinct primitive proposition.

Theorem 2.6: *Let $\varphi \in \mathcal{L}^{\approx}$ be a Boolean combination of approximate expressions not involving division. Let $\alpha(\varphi)$ be the propositional formula that results from replacing each approximate expression e in φ by a primitive proposition p_e. Then φ is valid over models of $\mathcal{L}^{\approx}$ iff $\alpha(\varphi)$ is propositionally valid.*

Thus, rather than considering validity, we concentrate on a different notion that we call *approximate entailment*.

3 Approximate Entailment

Given a knowledge base of approximate measurements, when should it entail a statement φ such as "John is approximately 1.9 meters tall"? We do not want to view φ as necessarily representing an actual measurement that was taken of John's height. Rather, we want it to be a useful working assumption. For example, if we are interested in buying John a suit, we

[1] In some cases we may know that tolerances are related in some way. For example, we may know that different measurements were taken with the same measuring device, and therefore have the same maximum error. This issue was dealt with in [GHK92] by explicitly subscripting $\approx$ and $\preceq$ so that each $\approx_i$ denotes a different approximate equality relation. For the sake of simplicity, we have chosen not to extend the logic to express relationships between tolerances.

may well be content with an approximate estimate of John's height. We do not expect formulas entailed by the knowledge base to be completely accurate. Moreover, we will rarely know exactly how accurate they are, since that depends on the accuracy of our initial measurements, which we do not typically know. However, we would like to have the property that the smaller the errors in the knowledge base, the smaller the errors in formulas entailed by the knowledge base. This is in the spirit of the standard ϵ-δ definition of limit.

Notice that it may not be possible to have all tolerances grow arbitrarily small simultaneously. For example, if our knowledge base consists of $(c \approx 1) \wedge (2c \approx 2.1)$, it is clear that both relevant tolerances cannot be arbitrarily small at the same time. We therefore introduce the concept of *minimal tolerance function*. Intuitively, this is one that chooses the smallest possible tolerances while still keeping the knowledge base consistent. We say that a tolerance function τ is *consistent with* KB if $KB[\tau]$ is satisfiable. We say that $\tau < \tau'$ for two tolerance functions τ, τ' if for all $e \in \mathcal{A}$, $\tau(e) \leq \tau'(e)$, and there exists some $e \in \mathcal{A}$ such that $\tau(e) < \tau'(e)$. We also define $\|\tau\|$ to be $\sup\{|\tau(e)| : e \in \mathcal{A}\}$. The tolerance function η is minimal for KB if it is in the closure of tolerance functions consistent with KB and there is no smaller tolerance function consistent with KB.

Definition 3.1: A tolerance function η is said to be *minimal for* KB if

1. for every $\epsilon > 0$, there exists a tolerance function τ consistent with KB such that $\|\tau - \eta\| \leq \epsilon$,
2. there does not exist another tolerance function τ such that $\tau < \eta$ and τ is consistent with KB.

Let $\Omega(KB)$ be the set of tolerance functions minimal for KB. ∎

Recall that τ_0 is the tolerance function that assigns tolerance 0 to all expressions. It is easy to see that if KB is consistent with equality, then τ_0 is the unique minimal tolerance function for KB. Thus, the notion of minimal tolerance function becomes interesting only for knowledge bases that are inconsistent with equality.

Example 3.2: The "inconsistent" height knowledge base KB' from the introduction, written formally in our language, is the conjunction: $(c_B \approx 1.8) \wedge (c_J \approx c_B + c_H/2) \wedge (c_H \approx 0.2) \wedge (c_J \approx 1.88)$, where c_B denotes Bill's height, c_J denotes John's height, and c_H denotes the height of a head. Let τ be a tolerance function such that $\tau(e) = 0$ for any irrelevant expression e (not one of the four above), and let $\tau_1, \tau_2, \tau_3, \tau_4$ denote the values assigned by τ to the four expressions above. Let $\vec{\tau}$ denote $(\tau_1, \ldots, \tau_4)$. It is easy to see that KB' is consistent iff $\tau_4 \geq 0.02 - \tau_1 - \tau_2 - \tau_3/2$. Thus, if $\vec{\tau} = (0, 0, 0, 0.01)$, then τ is not a minimal tolerance function for KB' because it violates condition 1. On the other hand, if $\vec{\tau} = (0.03, 0, 0, 0.01)$, then τ is not minimal because it violates condition 2: there exists a smaller tolerance function consistent with KB' that assigns $(0.02, 0, 0, 0)$ to the relevant expressions. This last tolerance function is in fact minimal for KB', as is the one that assigns $(0.01, 0, 0.02, 0)$ to the relevant expressions. And generally,

$$\Omega(KB') = \{\tau : \tau_4 = 0.02 - \tau_1 - \tau_2 - \tau_3/2, \ \tau(e) = 0 \text{ if } e \text{ is irrelevant}\}.$$ ∎

We now give an example of a formula KB for which the set of tolerances consistent with KB is not closed, and some minimal tolerance function is not consistent with KB.

Example 3.3: Let KB be $(c_1 \times c_2 \approx 1) \wedge (c_1 \approx 0)$, and let $\tau_1 = \tau(c_1 \times c_2 \approx 1)$ and $\tau_2 = \tau(c_1 \approx 0)$. It is clear that for any value of τ_1 and any $\tau_2 > 0$, $KB[\tau]$ is consistent. Therefore, one minimal tolerance function for KB is τ_0, which is inconsistent with KB. Note that the tolerance function τ' for which $\tau'_1 = 1$ and $\tau'_2 = 0$ is also consistent with KB (and therefore fulfills condition 1). And, although $\tau_0 < \tau'$, there does not exist a tolerance function $\tau < \tau'$ which is consistent with KB. Therefore, τ' is also a minimal tolerance function, and $\Omega(KB) = \{\tau_0, \tau'\}$. ∎

Remark 3.4: Although a minimal tolerance function for KB is not necessarily consistent with KB, it is the case that KB is satisfiable (i.e., some tolerance function is consistent with KB) iff $\Omega(KB) \neq \emptyset$. From this point on, we consider only satisfiable KB's. ∎

Using the concept of minimal tolerance functions, we can now define entailment. Before we give the formal definitions, we give a little intuition. We would certainly like a knowledge base of the form $c \approx 1$ to entail, say $2c \approx 2$. Recall from Example 2.5 that $(c \approx 1) \Rightarrow (2c \approx 2)$ is not valid. However, given a tolerance τ_1 for $c \approx 1$, we can clearly find a tolerance τ_2 for $2c \approx 2$ to make it true, namely, $2\tau_1$. This is the key idea in our notion of entailment. Roughly speaking, we want it to be the case that KB entails φ if, given a tolerance function τ that makes KB true, we can find a tolerance function τ' that makes φ true. Clearly we want to put some constraints on τ' (for otherwise from $c \approx 1$ we could infer $c \approx 2$). We require that the closer τ is to a minimal tolerance function, the closer τ' is to τ_0. This corresponds to the intuition we described earlier, that the smaller the errors in the knowledge base, the smaller the errors in the conclusion. Note that we do not try to describe *how* τ' must go to τ_0 as a function of how τ gets small.

We can now define the first of our two notions of entailment.

Definition 3.5: We say that KB *cautiously entails* φ, written $KB \mid\!\approx_c \varphi$, if for every minimal tolerance function $\eta \in \Omega(KB)$ there exists some function $f : \mathcal{T} \to \mathcal{T}$, and some $\epsilon > 0$ such that:

- for every tolerance function τ such that $\|\tau - \eta\| < \epsilon$, we have $KB[\tau] \models \varphi[f(\tau)]$,
- for every sequence $(\tau^n)_{n=1}^{\infty}$ such that $\lim_{n\to\infty} \tau^n = \eta$, we have $\lim_{n\to\infty} f(\tau^n) = \tau_0$. ∎

The reader might wonder why we insist that $f(\tau^n)$ converge to τ_0, rather than to a minimal tolerance function. The reason is that, even if we have a knowledge base that is inconsistent with equality, we want the formulas entailed by the knowledge base to be consistent with equality, since we want our conclusions to be useful working assumptions. Thus, if we have a knowledge base KB such as $(c \approx 0) \wedge (c \approx 0.1)$, then we do not want to conclude KB, as we would be able to do if we just required that $f(\tau^n)$ approach a minimal tolerance function rather than τ_0. As we shall show, although KB does *not* cautiously entail KB, it does cautiously entail $0 \preceq c \preceq 0.1$, which seems more reasonable.

Bold entailment replaces most of the universal quantifiers in the definition of cautious entailment by existential quantifiers.

Definition 3.6: We say that KB *boldly entails* φ, written $KB \mid\!\approx_b \varphi$, if either KB is unsatisfiable, or there exists some function $f : \mathcal{T} \to \mathcal{T}$, some minimal tolerance function $\eta \in \Omega(KB)$, and some decreasing sequence $(\tau^n)_{n=1}^{\infty}$ such that the following all hold:

- for all n, $KB[\tau^n] \models \varphi[f(\tau^n)]$,
- τ^n is consistent with KB for all n,
- $\lim_{n\to\infty} \tau^n = \eta$ and $\lim_{n\to\infty} f(\tau^n) = \tau_0$. ∎

Note that we restrict attention only to the tolerance functions τ^n in the first clause above, rather than requiring that this condition hold for all tolerance functions τ. The latter would also give us a reasonable notion of entailment. We chose our condition because it leads to a bolder notion of entailment; that is, it allows strictly more formulas to be entailed by a knowledge base. We have tried to make bold entailment as liberal as reasonably possible, while making cautious entailment as conservative as reasonably possible. Clearly other intermediate notions of entailment are possible.

As the names suggest, $\mid\!\approx_c \subseteq \mid\!\approx_b$ when viewed as relations on formulas. As Example 3.9 demonstrates, the containment is proper.

Proposition 3.7: *For $KB, \varphi \in \mathcal{L}^{\approx}$, if $KB \mid\!\approx_c \varphi$ then $KB \mid\!\approx_b \varphi$.*

We begin by showing a simple example of entailment.

Example 3.8: The consistent height knowledge base KB from the introduction, written formally in our language, is

$$(c_B \approx 1.8) \wedge (c_J \approx c_B + c_H/2) \wedge (c_H \approx 0.2) .$$

If we interpret $\approx$ as $=$, we can deduce that c_J, John's height, is 1.9 meters. As we might hope, using both cautious and bold entailment, KB entails that $c_J \approx 1.9$. Since $\mid\!\approx_c \subseteq \mid\!\approx_b$, it suffices to show this for cautious entailment. We proceed as follows: Since KB is consistent with equality, the only minimal tolerance function for KB is τ_0. Let τ_1, τ_2, τ_3 be the relevant coordinates of the tolerance function τ for KB. We choose $(f(\tau))(c_J \approx 1.9) = \tau_1 + \tau_2 + \tau_3/2$, and $f(\tau)(e) = 0$ for all other expressions $e \in \mathcal{E}$. Clearly, if $\lim_{n\to\infty} \tau^n = \tau_0$, then $\lim_{n\to\infty} f(\tau^n) = \tau_0$. Moreover, for any valuation v, if $|v(c_B) - 1.8| \leq \tau_1$, $|v(c_J) - v(c_B) - v(c_H)/2| \leq \tau_2$, and $|v(c_H) - 0.2| \leq \tau_3$, then $|v(c_J) - 1.9| \leq \tau_1 + \tau_3/2 + \tau_2 = f(\tau)(c_J \approx 1.9)$. Thus, if $(v, \tau) \models KB$, then $(v, f(\tau)) \models c_J \approx 1.9$. It follows that $KB \mid\!\approx_c c_J \approx 1.9$, as desired. ∎

The following example helps explain the difference between bold and cautious entailment. Intuitively, cautious entailment allows no unjustified default assumptions about the relationships between the tolerances in the knowledge base, whereas bold entailment allows arbitrary assumptions about these relationships.

Example 3.9: Consider the the knowledge base KB' from Example 3.2. This knowledge base is clearly inconsistent with equality, so using true equality we can deduce anything. What can we deduce using approximate entailment? Recall from Example 3.2 that

$$\Omega(KB') = \{\tau : \tau_4 = 0.02 - \tau_1 - \tau_2 - \tau_3/2\} .$$

It is easy to see that for any model (v, τ) consistent with KB' such that $\tau \in \Omega(KB')$, we must have $1.88 \leq v(c_J) \leq 1.9$; moreover, every value in the range $[1.88, 1.9]$ is attained in one of these models. It follows that $KB' \mid\!\approx_c 1.88 \preceq c_J \preceq 1.9$, and that we cannot get better bounds on c_J using cautious entailment. Thus, given contradictory information as to John's height, the cautious approach allows us to deduce only that the value of c_J is somewhere in the interval.

On the other hand, using the function f assigning $(f(\tau))(c_J \approx h) = \tau_4 - (0.02 - \tau_1 - \tau_2 - \tau_3/2)$ for any $1.88 \leq h \leq 1.9$ and zero elsewhere, we can deduce $KB' \mid\!\approx_b c_J \approx 1.88$, $KB' \mid\!\approx_b c_J \approx 1.89$, and in general, $KB' \mid\!\approx_b c_J \approx h$ for every $h \in [1.88, 1.9]$. Thus, the bold approach allows us to deduce any value for John's height in the permissible range; we may use any reasonable working assumption for John's height. ∎

The two approximate entailment relations are defined for a particular language $\mathcal{L}$ and a semantics for it. Clearly the definitions make perfect sense for a far richer language, for example, one with first-order quantification and with interactions between tolerances. (We remark that the decision procedure in

the next section also holds for this extended language, although our characterizations in Section 5 do not.) More interestingly, these relations can be extended to other semantics and other notions of satisfaction. The first clause in both of the approximate entailment definitions is based on the standard notion of satisfaction for precise formulas—$KB[\tau] \models \varphi[f(\tau)]$. We can replace the symbol $\models$ is this statement by a nonmonotonic notion, for example, or a probabilistic notion such as "holds with probability 1." This extension is explored further in Section 6.

4 A Decision Procedure for Approximate Entailment

In this section, we present decision procedures for the problems of deciding whether $KB \mathrel{|\!\sim\!\!\!\approx}_c \varphi$ and whether $KB \mathrel{|\!\sim\!\!\!\approx}_b \varphi$. Our decision procedures will be based on reducing these questions to the validity of certain formulas over the reals. We need first present the definition of a real closed field (see, for example, [Sho67]).

Definition 4.1: An *ordered field* is a field with a linear ordering $<$, where the field operations $+$ and $\times$ respect the ordering: that is, $x < y$ implies $x + z < y + z$, and if x, y are *positive* (where an element x is positive if $x > 0$) then so is $x \times y$. A *real closed field* is an ordered field where every positive element has a square root and every polynomial of odd degree has a root. ∎

Tarski [Tar51] showed that the theory of real closed fields coincides with the theory of the reals (under formulas containing only $+, \times, <, =$ and constants $0, 1, -1$). He also proved that the theory is decidable. Ben-Or, Kozen, and Reif [BKR86] extended this result to show that the complexity of the decision problem is exponential space.

When defining $\mathcal{L}^{\approx}$, we allowed a constant d_r for every real number r. Clearly, we cannot extend the decision procedure to formulas containing such constants. We therefore define a formula φ to be *rational* if for every constant d_r mentioned in φ, r is a rational number.

Theorem 4.2: *Given rational formulas $\varphi, KB \in \mathcal{L}^{\approx}$, we can in polynomial time construct formulas ψ_b and ψ_c over the vocabulary $\{0, 1, +, \times, <\}$ whose length[2] is linear in that of φ and KB, such that*

- $KB \mathrel{|\!\sim\!\!\!\approx}_c \varphi$ *iff* $\langle \mathbf{R}, 0, 1, +, \times \rangle \models \psi_c$,
- $KB \mathrel{|\!\sim\!\!\!\approx}_b \varphi$ *iff* $\langle \mathbf{R}, 0, 1, +, \times \rangle \models \psi_b$.

From the results of [Tar51] and [BKR86] mentioned above, we immediately get:

[2]The length of a rational formula φ is defined as the length of φ written as a string of symbols, where the length of d_q, where $q = a/b$ and a and b are integers, is the sum of the lengths of the binary representations of a and b.

Corollary 4.3: *For KB, φ rational formulas, the problem of deciding whether $KB \mathrel{|\!\sim\!\!\!\approx}_c \varphi$ (resp. $KB \mathrel{|\!\sim\!\!\!\approx}_b \varphi$) is in exponential space.*

5 Properties of Entailment

Although we have a decision procedure for approximate entailment, it does not give us much insight into the properties of these relations. In this section, we explore these properties in greater depth. We begin by showing the connection between approximate entailment and standard entailment in $\mathcal{L}$ (the logic of true equality). This gives a complete characterization for a large fragment of our language. We then use this characterization in order to relate approximate entailment to standard nonmononotonic formalisms.

5.1 Characterization

What kind of inferences can we make using our two notions of entailment? Most importantly, the inferences we make are always consistent with equality: they are always a subset of those we would obtain if we were to treat approximate equality as true equality. This is an important property; if it is consistent to interpret approximate equality as equality, then we do not want to conclude anything that would be inconsistent with this intepretation. One might also hope that for consistent knowledge bases, the converse also holds. This is in fact the case for bold entailment, and under certain conditions also for cautious entailment. Thus, we get all and only "reasonable" conclusions from knowledge bases consistent with equality. If this were the whole story, then there would be no need to introduce approximate equality at all. However, as we have already observed, some of the most interesting applications of approximate reasoning arise precisely when the knowledge base is inconsistent with equality. In this case, we do not want to be able to infer everything (as we could if we did not view equality as approximate). We will see, in fact, that the inferences that are lost are the "undesirable" ones.

Our goal is to characterize what we can infer in general. Roughly speaking, we want to show that a knowledge base KB cautiously entails φ if φ is true for *every* minimal tolerance function for KB; on the other hand, KB boldly entails φ if φ is true for *some* minimal tolerance function.

In order to relate entailment to the truth of formulas in $\mathcal{L}$, we first need a result that shows that entailment is, in some sense, continuous in the tolerance function.

Proposition 5.1: *Let KB and φ be two sentences in $\mathcal{L}^{\approx}$. Let τ^n be a decreasing sequence of tolerance functions such that $\lim_{n\to\infty} \tau^n = \eta$, and let f be a function such that $\lim_{n\to\infty} f(\tau^n) = \tau_0$. If, for every n, $KB[\tau^n] \models \varphi[f(\tau^n)]$, then $KB[\eta] \models \varphi[\tau_0]$.*

We can now give half of our desired characterization.

Theorem 5.2: *Suppose* $\varphi, KB \in \mathcal{L}^{\approx}$.

- *If* $KB \mid\!\approx_c \varphi$ *then for all* $\eta \in \Omega(KB)$, *we have* $KB[\eta] \models \varphi[\tau_0]$.
- *If* $KB \mid\!\approx_b \varphi$ *then for some* $\eta \in \Omega(KB)$, *we have* $KB[\eta] \models \varphi[\tau_0]$.

Note that if $KB[\tau_0]$ is consistent (i.e., the knowledge base is consistent with equality), then $\Omega(KB) = \{\tau_0\}$. Thus, as a corollary to Theorem 5.2, we get that our inferences are always consistent with equality:

Corollary 5.3: *If* $KB \mid\!\approx_b \varphi$ *(resp.,* $KB \mid\!\approx_c \varphi$*) then* $KB[\tau_0] \models \varphi[\tau_0]$.

This result is important, because it says that we cannot conclude anything (using either notion of entailment) that we could not have concluded by viewing approximate equality as equality.

The converses to Theorem 5.2 and Corollary 5.3 do not hold in general, nor do we want them to. If $KB[\eta]$ is inconsistent, then $KB[\eta] \models \varphi[\tau_0]$ for all φ, but we do not want to conclude $KB \mid\!\approx_b \varphi$ for all φ. In the case of bold entailment, such inconsistencies are the only problem.

Definition 5.4: We say KB is *min-consistent* if $KB[\eta]$ is consistent for all $\eta \in \Omega(KB)$. ∎

The knowledge-base in Example 3.3 is not min-consistent.

Theorem 5.5 : *If* KB *is min-consistent, then* $KB \mid\!\approx_b \varphi$ *iff* $KB[\eta] \models \varphi[\tau_0]$ *for some* $\eta \in \Omega(KB)$.[3]

This result characterizes $\mid\!\approx_b$ for min-consistent knowledge bases KB, and provides some justification for our calling this entailment "bold." It also lets us prove that, as long as our knowledge base is consistent with equality, then bold entailment lets us conclude precisely what we can conclude by viewing approximate equality as equality.

Corollary 5.6: *If* $KB[\tau_0]$ *is consistent, then* $KB[\tau_0] \models \varphi[\tau_0]$ *iff* $KB \mid\!\approx_b \varphi$.

We can prove results analogous to Theorem 5.5 and Corollary 5.6 for cautious entailment. However, we must place some restrictions on formulas, as the following examples show.

The first example shows that we cannot deal with formulas that use precise equality.

Example 5.7: Let KB be $c \approx 0$ and φ be $c = 0$. The knowledge base is consistent with equality; therefore $\Omega(KB) = \{\tau_0\}$. From $KB[\tau_0]$ we can infer $\varphi[\tau_0]$ (the two are, in fact, equivalent). However, it is not the case, nor do we want it to be, that $KB \mid\!\approx_c \varphi$. We point out that for the bold logic it is the case that $KB \mid\!\approx_b \varphi$.[4] ∎

The second example shows that we cannot deal with formulas that mention division either. Here the problem is the possibility of division by zero.

Example 5.8: Let KB be $c \approx 0$ and φ be $\neg(1/c \succeq 0)$. Clearly KB is consistent with equality. Moreover, it is easy to see that $c = 0 \models \neg(1/c \geq 0)$: according to our semantics $1/0 = \iota$ and $\iota \geq 0$ is false. However, it is also easy to see that KB does not cautiously entail φ: for all τ with $\tau(c \approx 0) > 0$, there is no choice of τ' such that $(c \approx 0)[\tau] \models (\neg(1/c \succeq 0))[\tau']$.[5] ∎

Therefore, in order to obtain the desired results, we cannot allow precise equality and division in KB or φ. But, as the following three examples show, further restrictions are necessary as well.

Example 5.9: Let KB be $((c \approx 1) \wedge (c+1 \not\approx 2)) \vee (c \approx 0)$. Clearly, $\Omega(KB) = \{\tau_0\}$. From $KB[\tau_0]$ we can infer $c = 0$, and thus KB boldly entails $c \approx 0$. We might also hope that KB cautiously entails $c \approx 0$, but this is not the case. Let $\tau_1 = \tau(c \approx 1)$ and $\tau_2 = \tau(c+1 \approx 2)$. For any τ such that $\tau_1 > \tau_2$, we can easily see that there are models $(v, \tau) \models KB$ such that $v(c)$ is within τ_1 of 1. Thus, $KB \not\mid\!\approx_c c \approx 0$. All we can deduce using cautious entailment is $KB \mid\!\approx_c (c \approx 1) \vee (c \approx 0)$. Note that this is, in fact, a reasonable conclusion if we are being cautious in assuming relationships between different tolerances. ∎

Example 5.10: It is easy to see that $c \approx 0 \mid\!\approx_c 2c \approx 0$; we simply define the function f in the definition of cautious entailment so that $f(\tau) = 2\tau$. Similarly, $c \approx 0 \mid\!\approx_c d_r \times c \approx 0$ for any r. However, if c' is another uninterpreted constant in the language, then $c \approx 0 \not\mid\!\approx_c c' \times c \approx 0$ (although $\Omega(c \approx 0) = \{\tau_0\}$ and $c = 0 \models c' \times c = 0$). The reason is that for every tolerance function τ such that $\tau(c \approx 0) > 0$ and every constant $B > 0$, there is some model (v, τ) such that $v(c' \times c) > B$. We cannot place an *a priori* bound on the tolerance required for $c' \times c \approx 0$ in terms of $\tau(c \approx 0)$. ∎

Example 5.11: Let KB be $(c \approx 0) \vee (c \approx 1)$ and φ be

$$[(c \approx 0) \wedge ((d \not\approx 0) \vee (d \approx c(c-1)))] \quad \vee$$
$$[(c \approx 1) \wedge ((d \approx 0) \vee (d \not\approx c(c-1)))]$$

[3] If we were to require that the range of a tolerance function is $(0, \infty)$, so that τ_0 is not a a legal tolerance function, this theorem would not hold. Additional assumptions, similar to the well-behavedness assumptions below, would be necessary.

[4] In fact, it is easy to see that if KB is min-consistent, and $KB \mid\!\approx_b \varphi$, then $KB \mid\!\approx_b \varphi[\tau_0]$.

[5] This problem might seem to be an artifact of our particular semantics for division by zero. However, similar examples can be constructed for other choices.

Under true equality, the second conjunct in each disjunct of φ is implied by the first, so that $KB[\tau_0] \models \varphi[\tau_0]$. Under approximate equality, on the other hand, the situation is very different: As we show in the full paper, $KB \not\approx_c \varphi$. To see why at an intuitive level, recall that the first clause in the definition of cautious entailment requires us to find a function f such that $KB[\tau] \models \varphi[f(\tau)]$, for every τ small enough. For any τ small enough, we can easily find v and v' such that $(v, \tau) \models c \approx 0$ and $(v', \tau) \models c \approx 1$. We must therefore have $(v, f(\tau)) \models (d \not\approx 0) \vee (d \approx c(c-1))$ and $(v', f(\tau)) \models (d \approx 0) \vee (d \not\approx c(c-1))$. However, we cannot define $f(\tau)(d \approx 0)$ and $f(\tau)(d \approx c(c-1))$ so that both implications hold. Essentially, the problem is that the two expressions $d \approx 0$ and $d \approx c(c-1)$ appear both negated and unnegated in φ, inducing interactions between the two tolerances. ∎

These three examples essentially characterize the reasons why we do not get an analogue to Theorem 5.5 for cautious entailment, even for the restricted language. To make this precise, we need some definitions.

Definition 5.12: The constant c is said to be *bounded by* KB if KB implies $d_r \preceq c \preceq d_{r'}$ for some constants r and r'. ∎

Notice that the constant c' is not bounded by the KB $c \approx 0$ in Example 5.10.

Definition 5.13: Let φ be a formula in $\mathcal{L}^{\approx}$. We say that an expression $e \in \mathcal{A}$ *appears positively* (resp., *appears negatively*) in φ if there is an instance of e which is in the scope of an even (resp. odd) number of negations. We say that φ is *strictly independent* if there is no expression e that appears both positively and negatively in φ. ∎

Note that the formula φ in Example 5.11 is not strictly independent. Strict independence might seem, at first glance, to be a harsh restriction. But this is not the case. Consider, for example, $(c \approx 0) \wedge (c \not\approx 0)$. This formula is not strictly independent, but the almost identical formula $(c \approx 0) \wedge (c + 0 \not\approx 0)$ is. In general, it is simple to transform any φ to a strictly independent formula φ', that, apart from interactions between different tolerances, is equivalent.

Definition 5.14: The pair KB, φ is said to be *well-behaved* if

- neither KB nor φ contain any division operations or precise equality expressions,
- KB is negation free,
- all the constants in φ and KB are bounded by KB,
- φ is strictly independent. ∎

How reasonable is the assumption of well-behavedness? Since we are mainly interested in knowledge bases with approximate information, not allowing precise equality in this context does not seem unduly restrictive. Not allowing division is, of course, a nontrivial restriction, but still seems to cover many interesting examples. As we have seen, strict independence is a very mild restriction. Although for general knowledge bases we want negations, in this case we are reasoning only about quantitative relations among measured quantities and numerical constants. It seems reasonable to expect that the information we have in such a knowledge base would be positive. Finally, although the assumption that all constants that appear be bounded may seem restrictive, note that the bounds can be arbitrarily large. In practice, we often do have *some* bounds on the size of constants, perhaps not very precise. For example, if we are talking about heights of people, we surely have a lower bound of 0 and an upper bound of 3 meters. It should not hurt to add such bounds to the knowledge base. We therefore believe that in practice, knowledge bases will often be well-behaved (or can easily be made so). As we now show, well-behavedness is sufficient to guarantee that we avoid the problems in the examples above, as well as other difficulties.

Assuming well-behavedness, we can now prove that if an assertion holds at some minimal point, then it also holds for any sequence tending to that point.

Proposition 5.15: *Let KB, φ be well-behaved, and let η be a tolerance function such that $KB[\eta]$ is satisfiable. If $KB[\eta] \models \varphi[\tau_0]$, then there exists a function f and some $\epsilon > 0$ such that for all τ such that $\|\tau - \eta\| < \epsilon$, we have $KB[\tau] \models \varphi[f(\tau)]$, and for all sequences τ^n such that $\lim_{n\to\infty} \tau^n = \eta$, we have $\lim_{n\to\infty} f(\tau^n) = \tau_0$.*

We can now prove the following analogues to Theorem 5.5 and Corollary 5.6.

Theorem 5.16: *If the pair KB, φ is well-behaved, then $KB \approx_c \varphi$ iff $KB[\eta] \models \varphi[\tau_0]$ for all $\eta \in \Omega(KB)$.*

Corollary 5.17: *If KB, φ are well-behaved and $KB[\tau_0]$ is consistent, then $KB \approx_c \varphi$ iff $KB[\tau_0] \models \varphi[\tau_0]$.*

We can also show that the assumptions of Theorem 5.16 are stronger than those of Theorem 5.5:

Lemma 5.18: *If KB is bounded and negation-free then KB is min-consistent.*

We do not have an elegant characterization of bold entailment in the case where KB is not min-consistent, nor for cautious entailment in the case where φ and KB are not well-behaved. As our various examples show, we still get reasonable entailments even when these conditions are not met.

5.2 Preference semantics

Our characterization theorems emphasize the importance of minimal tolerance functions. Tolerance functions consistent with a knowledge base KB are possible combinations of measurement errors that could have led to the formation of KB. Since we prefer to believe that the errors made were as small as possible, we can view the ordering $<$ on tolerance functions as defining a preference relation on tolerance functions, in the spirit of [Sho87]. Therefore, for the fragment of our language for which the characterization theorems (Theorems 5.5 and 5.16) hold, approximate entailment reduces to reasoning in the preferred models. Using these results, we can now show that for well-behaved formulas, cautious entailment is closely related to Shoham's notion of *preferential entailment* [Sho87].

Definition 5.19: For two augmented models $M = (v, \tau)$ and $M' = (v', \tau')$, $M < M'$ if $\tau < \tau'$. The augmented model M is a *preferred model* of KB if $M \models KB$ and there is no other augmented model M' such that $M' < M$ and $M' \models KB$. KB *preferentially entails* φ, written $KB \models_< \varphi$, if for any preferred model M of KB, $M \models \varphi[\tau_0]$.[6] ∎

Theorem 5.20: *For well-behaved KB, φ the following are true:*

- *An augmented model $M = (v, \eta)$ of KB is a preferred model of KB iff $\eta \in \Omega(KB)$.*
- *$KB \mid\!\approx_c \varphi$ iff $KB \models_< \varphi$.*

We can view minimal tolerance functions as frames of mind, and models v for $\mathcal{L}$ as possible worlds. In the frame of mind corresponding to the minimal tolerance function η, the agent believes φ if $KB[\eta] \models \varphi[\tau_0]$. In the cautious approach, the agent believes only deductions made in all frames of mind. In the bold approach, the agent believes deductions made in any frame of mind. Note, however, that the agent can believe φ in one frame of mind and ψ in another, while not believing $\varphi \wedge \psi$ in any frame of mind. Therefore, it is possible that $KB \mid\!\approx_b \varphi$, $KB \mid\!\approx_b \psi$, while $KB \not\mid\!\approx_b \varphi \wedge \psi$.

Using the terminology of [KLM90], what this discussion has shown is that bold entailment is not closed under the *And* rule. We have also shown that neither bold nor cautious entailment is closed under the *Reflexivity* rule: it is not necessarily the case that $KB \mid\!\approx_b KB$ or that $KB \mid\!\approx_c KB$. Our characterization theorems suggest why this should be so: formulas on the left-hand side of $\mid\!\approx_b$ or $\mid\!\approx_c$ are evaluated with respect to minimal tolerance functions; formulas on the right-hand side are evaluated with respect to τ_0. Therefore, for KB inconsistent with equality, KB will not entail itself (nor would we want it to). Several examples have demonstrated that cautious entailment is nonmonotonic: adding information to the knowledge base can cause inferences to be lost. Interestingly, bold entailment is monotonic for a large fragment of the language. Both $\mid\!\approx_b$ and $\mid\!\approx_c$ are closed under most of the other rules suggested in [KLM90] (under certain restrictions such as strict independence). We discuss this issue in more detail in the full paper.

[6] Shoham's notation for M preferred to M' is $M' < M$. We choose to represent preference as $M < M'$ in order to maintain consistency with the notation for tolerance functions. Moreover, this is not quite Shoham's definition of preferential entailment. The exact analogue of Shoham's definition would have $M \models \varphi$, not $M \models \varphi[\tau_0]$.

6 Applications

6.1 Error propagation

So far, we have only looked at very simple examples, with two or three quantities of interest, and very few interactions between them. In real-world situations, there will be many different quantities, each of which will be relevant in a variety of computations. This can cause complex interactions, as shown in the following simple example.

Example 6.1: Consider a robot operating in a blocks world, whose primitive actions are: grasp (g), ungrasp (u), and move arm (m). Suppose that the robot has estimates on how long these tasks take. Such estimates are clearly useful in the context of planning. Let $c_g \approx 2$, $c_u \approx 1.5$, and $c_m \approx 5$ be the estimates for the time taken by these three actions, respectively. Assume that a particular plan r (raise block) requires one grasp and one move action, so that the robot estimate that $c_r \approx c_g + c_m$, whereas plan l (lower block), requires one move and one ungrasp, so that $c_l \approx c_m + c_u$. The robot can deduce that $KB \mid\!\approx_c c_r \approx 7 \wedge c_l \approx 6.5$. Now suppose that the robot executes plan r, measures the time it actually takes, and discovers that it takes 7.5 seconds. The robot would like to use this measurement as a new approximation for how long this plan usually takes. It therefore adds $c_r \approx 7.5$ to KB, obtaining KB'. The robot can now conclude that $KB' \mid\!\approx_b c_g \approx 2.5 \wedge c_l \approx 6.5$; this corresponds to the case that the mistaken estimate was for action g, and therefore the time for plan l is unaffected. Alternatively, the robot can conclude $KB' \mid\!\approx_b (c_g \approx 2) \wedge (c_m \approx 5.5) \wedge (c_l \approx 7)$, corresponding to the case that the mistaken estimate was for action m. Yet another alternative is that $KB' \mid\!\approx_b (c_g \approx 2) \wedge (c_m \approx 5) \wedge (c_l \approx 6.5)$, corresponding to the case that the cause of the discrepancy was overhead in plan r. Any intermediate assignment of errors is also possible. Note that each of these alternatives corresponds to a scenario that "explains" the discrepancy between the estimated time for the plan and the actual time for the plan by assuming that the estimates were as correct as possible, and making the minimal change required to account

for the discrepancy. If the robot is not willing to leap to any of these conclusions, then it could use cautious entailment, and obtain ranges for the estimated time for each action and plan. ∎

We see that the process of considering the different causes for the discrepancy, and deducing from those how the times for different plans could be affected, is done automatically by approximate entailment. Essentially, errors are propagated back to their possible sources, and then forward to their logical conclusions. This type of reasoning is useful in many other applications. For example, it arises in complex numerical computations, where each subroutine can introduce errors (such as rounding errors), which then propagate in many ways, affecting more than one result.

6.2 Measurement theory

The problems of inexact measurement and numerical inaccuracies have been extensively investigated in the field of measurement theory. While there are many points of commonality between our approach and measurement theory, there are also some significant differences. Measurement theory investigates the issue from an axiomatic standpoint. Their measurement data typically contains relative observations about the objects being measured. For example, if we are measuring the heights of people, we may observe that John is taller than Bill. The general theory attempts to find axioms guaranteeing that numbers that "satisfy" the observations in an appropriate sense can be assigned to the objects. Thus, they start from axioms, rather than models, as we do.

When dealing with inexact measurement, the problems encountered typically involve *intransitivities*. Consider the following classic example [LR57]:

Example 6.2: Let c_n denote a standard cup of coffee that contains n granules of sugar. The agent cannot differentiate between c_n and c_{n+1} by taste; therefore, its KB will contain $c_n \approx c_{n+1}$ for every n. However, for some m, the agent will be able to tell that c_m is sweeter than c_1, so KB will contain $c_1 \prec c_m$. ∎

Intuitively, from the point of view of measurement theory, this problem arises because each measurable quantity c_i has a "true value," and an interval around it that the agent cannot differentiate from the true value. It is shown [SKLT89] that if the $\prec$ relation satisfies certain axioms, then we can find an assignment v and a *threshold function* δ such that taking $c_i \approx c_j$ iff $v(c_i) \in [v(c_j) - \delta(c_j), v(c_j) + \delta(c_j)]$ is consistent with all the observations. Such a pair (v, δ) is called a *threshold representation*.

This shows another key difference between our approach and measurement theory. Measurement theory associates the "tolerance" δ with a quantity such as c_i, whereas we associate a tolerance with a measurement such as $c_i \approx c_j$, viewing each measurement as having its own (independent) uncertainty. However, we can capture the measurement theory notion of tolerance in our approach if we put additional constraints on our tolerance function τ. In particular, we could require for each variable c_i, the tolerances of all comparisons involving c_i are the same; i.e., for all j and k, we could require that $\tau(c_k \approx c_i) = \tau(c_j \approx c_i)$. Thus, a threshold representation corresponds to an augmented model with some additional constraints on the tolerance function, so axioms that guarantee the existence of a threshold representation also guarantee the existence of such an augmented model.

6.3 Probabilistic entailment

Another important type of numerical information is probabilistic knowledge. An agent may frequently use probabilities to deal with its uncertainty about the truth of various sentences. For example, in [Nil86], Nilsson suggests a framework for probabilistic logic, which, for the case of propositional logic, is essentially as follows.

Consider a finite propositional language over the propositions $p_1, \ldots, p_k$. There are $K = 2^k$ truth assignments, or worlds, for this language; let us denote them by $w_1, \ldots, w_K$. Given a probability distribution π over the K worlds, we define the probability of a propositional sentence α to be

$$\pi(\alpha) = \sum_{w_i \models \alpha} \pi(w_i) \; . \tag{1}$$

The agent's knowledge base consists of a set of constraints on the probabilities of different sentences; for example, $\pi((p_1 \vee p_3) \wedge \neg p_6) \leq 0.4$. Using *probabilistic entailment*, the agent deduces constraints on the probability of a sentence α from the probabilistic constraints in the knowledge base.

In our framework, we define the constant c_i to be the probability $\pi(w_i)$. Any constraint on the probability of sentences can be replaced by a constraint referring only to the constants c_i, using Equation (1). Thus, the problem of probabilistic entailment can be expressed in $\mathcal{L}$. But what happens if the numbers appearing in the probabilistic constraints are not known to be precise? As Nilsson points out, "just as it is possible to assign inconsistent *true-false* truth values to sentences, it is also possible to assign them inconsistent probabilities." He suggests the heuristic, based on a geometric interpretation, of moving to a "nearby" consistent probability distribution. This is, in fact, the effect of using approximate equality (instead of true equality) to represent the constraints, and using our framework for approximate entailment.

The same framework for assigning probabilities to propositional sentences is also used in Pearl's ϵ-

semantics [Pea88]. The goal of ϵ-semantics is to provide probabilistic semantics for default reasoning—defaults of the type "birds fly" are interpreted as meaning "almost all birds fly," and are given semantics using the conditional probability of the "fly" given "bird" (where "fly" and "bird" are propositions in the language). However, the statement $\pi(fly \wedge bird)/\pi(bird) = 1$ does not accurately represent the meaning of the sentence "almost all birds fly," since it is inconsistent with the existence of non-flying birds. Pearl's solution to the problem is essentially equivalent to representing that sentence as $\pi(fly \wedge bird)/\pi(bird) \approx 1$ (using the transformation of π into constants c_i as described above).[7] Using this representation, Pearl defines a notion of *ϵ-entailment*, written $\models_\epsilon$. As we now show, our notion of cautious entailment agrees with ϵ-entailment, except when the knowledge base is declared inconsistent by the ϵ-semantics approach. In this case, cautious entailment can tolerate the inconsistency and still provide reasonable answers.

Theorem 6.3: *Let Δ be a set of defaults and α a sentence in the language of ϵ-semantics. Let KB and φ be their respective analogues in $\mathcal{L}^{\approx}$, using the transformation described above. Then if $\Omega(KB) = \{\tau_0\}$, then $\Delta \models_\epsilon \alpha$ iff $KB \mid\!\approx_c \varphi$.*

Note that this equivalence holds only if the knowledge base is consistent arbitrarily close to τ_0. If this is not the case, then ϵ-semantics would declare the knowledge base to be inconsistent, allowing arbitrary deductions. Consider, for example, the following variant (from [Pea88]) of the famous *lottery paradox* [Kyb61].

Example 6.4: A large number of people buy tickets for a lottery that will have a single winner. Let c_i represent the probability that person i will win. The probability that any one person will win is very low. Therefore, we might choose to represent our knowledge in ϵ-semantics using the statement $c_i \approx 0$. However, we know that one person will certainly win, leading to the conclusion $c_1 + c_2 + \ldots + c_N = 1$, where N is the number of people participating in the lottery. The resulting knowledge base is inconsistent in the ϵ-semantics framework. However, using our approach, this inconsistency is avoided, and the knowledge base can be used for making inferences. ∎

Both Nilsson and Pearl have suggested specific nonmonotonic variants of their basic logics. Our formalism easily extends to encompass these proposals. Both proposals use the same basic idea: instead of looking at all probability distributions over worlds consistent with the constraints, one should look at one particular "special" one—the probability distribution satisfying the constraints that has *maximum entropy* (see [GMP90] for more details). This leads to nonmonotonic notions of inferences: we leap to the conclusions sanctioned by the distribution of maximum entropy consistent with our information, although they may not be sanctioned by other distributions consistent with our information. We can easily extend our framework to deal with nonmonotonic inferences, and thus capture these nonmonotonic approaches. As before, let $KB, \varphi \in \mathcal{L}^{\approx}$ represent the knowledge base and desired conclusion in our framework. Recall that the first condition in the definition of cautious entailment states that, for every τ, $KB[\tau] \models \varphi[f(\tau)]$. Until now, we used $\models$ to denote the standard notion of entailment. Instead, we can replace $\models$ by $\models_{ME}$, where $\models_{ME}$ allows any inferences which hold in the model v of $KB[\tau]$ having maximum entropy (where we now view v as a probability assignment, so that talking about its entropy makes sense), leaving the remainder of the definition unchanged. The key point here is that the choice of inference rule is completely orthogonal to our treatment of approximate equality. Our approach can be applied to any notion of inference rule, to convert a logic for reasoning about equality to one for reasoning about approximate equality.

A final example of the generality of this framework uses yet another language and inference mechanism. In [GHK92], we present a technique which deduces *degrees of belief* from a first-order knowledge base augmented with statistical information about the domain. That is, given such a knowledge base KB and a formula φ, we define the notion of the degree of belief in φ given KB, denoted $\Pr_\infty(\varphi|KB)$ (see [GHK92] for details). The statistical information has the form "the proportion of flyers among birds is 90%;" this type of information is usually based on some sort of statistical sampling, and is therefore only approximate. Moreover, as pointed out in [GHK92], if we were to interpret the statistical statement above as being precisely true, we would deduce that the number of birds is a multiple of 10, an inference which is surely undesirable. Therefore, approximate equality rather than precise equality is used in [GHK92]. However, there is no analogue to (cautious or bold) entailment. Rather, the approach of [GHK92] can be viewed as using an analogue to validity (for an appropriate nonstandard notion of $\models$). Since, as we have seen, very few interesting deductions regarding approximate inference can be made using validity, not much can be deduced if we have approximate equality in both φ and KB. As an example of this phenomenon, suppose KB is "10% of birds are yellow, 20% of birds are green, and no birds are both yellow and green," and φ is "30% of birds are yellow or green." Using the approach of [GHK92], we cannot deduce $\Pr_\infty(\varphi|KB) = 1$. Moreover, this approach could not deal with inconsistent numerical

[7] We cannot represent this formula as $\pi(fly \wedge bird) \approx \pi(bird)$. Assume, for example, that we also knew that most things are not birds, that is, $\pi(bird) \approx 0$. Then, for any τ such that $\tau(\pi(bird) \approx 0) \leq \tau(\pi(fly \wedge bird) \approx \pi(bird))$, the assertion $\pi(fly \wedge bird) \approx \pi(bird)$ would be trivially true. This is not the case for the original assertion above.

information in KB. By using the approach outlined in this paper, one could define both a bold and a cautious version of Pr_∞, and deal with these issues in a satisfactory way.

7 Conclusions

We have presented a logic for approximate reasoning, and defined two notions of approximate entailment used to make default deductions from an imprecise knowledge base. One might ask why we should bother designing a new logic, rather than using say, a variant of *relevance logic* [AB75], *fuzzy logic* [Zad75], or any one of a number of nonmonotonic logics. Each of these logics has some properties that we view as desirable in our setting. The use of relevance logic would block the deduction of arbitrary formulas from an inconsistent knowledge base. Fuzzy logic would allow us to express the notion of approximate equality (although in a way that is very different from that captured by our semantics). Nonmonotonic logics allow the type of nonmonotonic behavior we want in the height knowledge base mentioned above. However, since these logics were not designed specifically to handle approximate measurement, none of them can capture all the situations in which we are interested. Threshold representations and semantics based on intervals can be used to directly express approximate quantities. In fact, Parikh's [Par83] theory of *vague reals* is a logical framework for doing this. However, as we explained in Section 6.2, approximate quantities differ from approximate measurements. Moreover, Parikh's framework requires observations to be given in terms of ranges rather than exact numbers; this is not always feasible.

Our logic has many interesting and intuitive properties. We concentrated on demonstrating these properties for a particular language (the language of arithmetic), and for the classical notion of entailment. However, we also showed variants of our logic for probabilistic and even nonmonotonic logics. We view this logic as providing a general and coherent framework for dealing with approximate information and the numerical inconsistencies that usually accompany it.

Acknowledgements

The authors thank Adam Grove for numerous helpful discussions, and Ron Fagin and Moshe Vardi for useful comments on a draft of the paper. This research was sponsored in part by the Air Force Office of Scientific Research (AFSC), under Contract F49620-91-C-0080.

References

[AB75] A. Anderson and N. D. Belnap. *Entailment: The Logic of Relevance and Necessity*. Princeton University Press, Princeton, NJ, 1975.

[BKR86] M. Ben-Or, D. Kozen, and J. Reif. The complexity of elementary algebra and geometry. *Journal of Computer and System Sciences*, 32(1):251–264, 1986.

[GHK92] A. J. Grove, J. Y. Halpern, and D. Koller. Random worlds and maximum entropy. In *Proc. 7th IEEE Symp. on Logic in Computer Science*, 1992.

[GMP90] M. Goldszmidt, P. Morris, and J. Pearl. A maximum entropy approach to nonmonotonic reasoning. In *Proc. AAAI National Conference*, pages 646–652, 1990.

[KLM90] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.

[Kyb61] H. E. Kyburg, Jr. *Probability and the Logic of Rational Belief*. Wesleyan University Press, Middleton, CT, 1961.

[LR57] R. D. Luce and H. Raiffa. *Games and Decisions – Introduction and Critical Survey*. John Wiley & Sons, 1957.

[Nil86] N. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.

[Par83] R. Parikh. The problem of vague predicates. In R. S. Cohen and M. W. Wartofsky, editors, *Language, Logic, and Method*, pages 241–261. D. Reidel Publishing Company, 1983.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, CA, 1988.

[Sho67] J. R. Shoenfield. *Mathematical Logic*. Addison-Wesley, Reading, MA, 1967.

[Sho87] Y. Shoham. A semantical approach to nonmonotonic logics. In *Proc. 2nd IEEE Symp. on Logic in Computer Science*, pages 275–279, 1987.

[SKLT'89] P. Suppes, D. M. Krantz, R. D. Luce, and A. Tversky. *Foundations of Measurement*, volume 2. Academic Press, Inc., 1989.

[Tar51] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. Univ. of California Press, 2nd edition, 1951.

[Zad75] L. A. Zadeh. Fuzzy logics and approximate reasoning. *Synthese*, 30:407–428, 1975.

A Spatial Logic based on Regions and Connection

David A. Randell, Zhan Cui and Anthony G. Cohn

Division of Artificial Intelligence
School of Computer Studies
Leeds University
Leeds, LS2 9JT, England
{dr, cui, agc}@scs.leeds.ac.uk

Abstract

We describe an interval logic for reasoning about space. The logic simplifies an earlier theory developed by Randell and Cohn, and that of Clarke upon which the former was based. The theory supports a simpler ontology, has fewer defined functions and relations, yet does not suffer in terms of its useful expressiveness. An axiomatisation of the new theory and a comparison with the two original theories is given.

1 Introduction

The use of interval logics for the representation of time are well known in AI research - see for example Allen (1984) and Allen and Hayes (1985) although their development and history extends back much further in philosophical literature, see for example Hamblin (1967, 1971). However, despite the intuitive connection that can be drawn between space and time in terms of such logics, until fairly recently, little work in AI has centred on the development and use of interval logics for space.

We describe an interval logic that can be used to reason about space. The similarity of the title with Clarke's (1981) paper 'A calculus of individuals based on 'connection' ', is not accidental. In Randell and Cohn (1989, 1992) and in Randell (1991) we used Clarke's theory as a foundation to build a theory that supported some basic intuitions about the nature of space, time and processes. Although this theory is formally sound, in use, we found some features of both Clarke's theory and our own proved problematic. This led to a re-evaluation of the original theory which is presented below.

The structure of the rest of this paper is as follows. In sections 2 and 3, we give a brief overview of the original theory, and point out the various problems we encountered that led to the development of the revised theory. In section 4 we give the axiomatised theory, drawing out the contrasts with the original theory. In section 5 we discuss the implications of introducing atomic regions into the theory, and in section 6 we discuss related and further work.

2 Overview of the original spatial theory

The original theory (see Randell and Cohn 1989, 1992 and Randell 1991) is based upon Clarke's (1981, 1985) calculus of individuals based on "connection" and is expressed in the many sorted logic LLAMA - see Cohn (1987).

The ontological primitives of the theory include physical objects, regions and other sets of entities. These and other specialisations of these primitive sets of entities are all treated as sorts in the theory and are subsequently embedded in a complete Boolean lattice, forming a sort hierarchy. However, given the scope of this paper, we shall limit the overview of the original theory to that which applies to space, while reminding the reader that what follows is but a small part of a much larger theory.

The basic part of the theory assumes a primitive dyadic relation: $C(x, y)$ read as 'x connects with y' which is defined on regions; this is axiomatised to be reflexive and symmetric. In terms of points incident in regions, $C(x, y)$ holds when regions x and y share a common point. Using the relation $C(x, y)$, a basic set of dyadic relations are defined. These relations describe differing degrees of connection between regions from being disconnected, to being externally connected, allowing partial overlap, one region being a tangential part of the other, or a nontangential part, and so on. All degrees of connection from being externally connected to sharing mutual parts and thus being identical are formally defined.

The theory also supports a set of functions that define the Boolean composition of regions, and a set of topological functions that allow for the explicit repre-

sentation of the interior, the closure and the exterior of particular regions. We also extend the basic theory outlined by Clarke by including a further set of dyadic relations that are used to describe regions being either inside, partially inside, or outside another. Several variants are defined.

The spatial part of the theory represents but a part of a much larger theory, which is now briefly covered. The theory enables the user to describe states, events and processes. For this a set of ternary relations are introduced that enable one to relate pairs of bodies using the dyadic relations mentioned above over time. These are subsequently used to create a set of envisioning axioms in the general theory, which impose constraints upon the manner in which bodies can vary in their degree of connection over time. These form the basis of processes described in the theory, where processes are described in terms of stipulated sequences of direct topological transitions allowed between sets of objects. These processes can either be reasoned about using a direct theorem proving implementation of the theory, or by using a simulation program - see Cui, Cohn and Randell (1992) for further details.

3 Problems

There are several problems that have arisen during our course of research using the original theory. These can be conveniently classified under three distinct, but related headings: conceptual, pragmatic and computational. We shall discuss these in turn.

A common question asked of us concerning the original theory was why we needed to introduce the topological distinctions between the types of regions assumed by the general theory. From the naive point of view, it seemed odd to have open, semi-open and closed regions as a model for regions. This point simply reflects a general concern made by writers in both philosophy and science, that a remoteness exists between the facts of actual observation and the descriptive language used. In Philosophy, this has resulted in a strong interest in developing languages with a clear primitive observational or phenomenal content; languages that can be directly related to the world around us (Hamblin 1971). For example, in terms of content, it seems odd that two regions can be distinct, but that each occupies the same amount of space, as in the case where we take an open region, and its closure.[1] Moreover, given the explicit use of different types of topological regions for describing space, we have the odd result that if a body maps to a closed region of space (which is a natural association), its complement is open, and that if we consider a body which is broken into two parts, then we have a problem how to split the regions so formed, since any closed interval that is split into two must have a semi-open part, and which is which?[2] From the standpoint of our naive understanding of the world, this topological structure is arguably too rich for our purposes, and in any case appearing in this formal theory, it poses some deep conceptual problems.

Given the choice between two possible theories used for formally representing space, the ease by which a person can understand and use the theory must be taken into account. The basic part of the original theory, concerning regions required the user to be familiar with general topology, both in order to understand the theory, and for any person wishing to extend the theory. We thought this restriction could be eased, but this required a change in the ontology of regions assumed by the original theory, and changes in the extant axiomatisation.

Clarke's (1981, 1985) calculus of individuals is simply presented as an unsorted first order theory, and as such, questions of implementation are understandably not addressed. However, in our case, we had to keep implementational and efficiency questions to the fore. We decided to use a sorted logic, since their effectiveness in reducing the search space for many problems in automated reasoning is well known. Also we wanted to keep our syntax as clear as possible, by absorbing all the monadic predicates in the theory and pushing these into the sortal part of the logic. We originally decided to implement our original theory using Cohn's (1987) sorted logic LLAMA, but this required much groundwork first, since the logic requires the user to first specify the positions of the sorts in the sort hierarchy[3]. This required us to first prove in the sorted theory, for any two potential sorts (being the monadic predicates of the unsorted theory), whether they were disjoint or whether one subsumed the other. This proved to be a particularly difficult and tedious task, which was made especially difficult given the spartan nature of the primitives used in the theory, which meant even basic theorems could prove difficult to tease out. Part of the problem simply lay in the number of potential subsorts, of the sort REGION, we had defined, and again this in part stemmed from the topological basis of the theory stemming from Clarke's theory[4].

[1] A very clear example of this was suggested by Antony Galton, who pointed out that the northern hemisphere, with or without the equator includes the same amount of regional space - the former being a closed region, the latter an open one

[2] It is interesting to note too that the same difficulties for space also arise in the temporal model, for example, deciding whether the order of intervals should be either ( ], or [ ). See Galton (1990) for further discussion.

[3] However, more recently, LLAMA has been relaxed and only partial sort information need be specified - see Cohn(1992).

[4] By having three kinds of regions (open, closed, semi-open), the number of sorts was immediately increased threefold.

Taking all these factors into account we eventually decided to investigate how the theory could be simplified; this is presented below.

4 The new theory

The new theory, like the original theory, is based upon Clarke's calculus of individuals based on "connection" and again is expressed in the many sorted logic LLAMA. Reasons of space mean that we cannot give full details of the sorted logic assumed below. However, for the purposes of reading this paper, all the reader should bear in mind is that LLAMA allows arbitrary ad hoc polymorphism, and that the variables are not explicitly typed, but that their associated sorts are derived implicitly from their argument positions in specified formulae. We will occasionally highlight certain sortal restrictions; in this case sorts in the theory will be indicated by strings of upper case letters, e.g. REGION, SPATIAL and NULL.

The ontological primitives of the (extended) new theory include physical objects, regions and other sets of entities. These and other specialisations of these primitive sets of entities are all treated as sorts in the theory and are subsequently embedded in a complete Boolean lattice, forming a sort hierarchy. However, here, by restricting ourselves to a theory describing space, we shall only concern ourselves with those sorts that specialise the sort SPATIAL.

Regions in the theory support either a spatial or temporal interpretation. Informally, these regions may be thought to be potentially infinite in number, and any degree of connection between them is allowed in the intended model, from external contact to identity in terms of mutually shared parts.

The basic part of the formalism assumes one primitive dyadic relation: $C(x, y)$ read as 'x connects with y'. For the basic part of the theory, the individuals can be interpreted as either spatial or temporal regions, but as we are describing a theory for space, a spatial interpretation is assumed in the pictorial model we give in Figure 1. The relation $C(x, y)$ is reflexive and symmetric. We can give a topological model to interpret the theory, namely that $C(x, y)$ holds when the topological closures of regions x and y share a common point.[5] Two axioms are introduced.

$\forall x C(x, x)$
$\forall x y[C(x, y) \rightarrow C(y, x)]$

Using $C(x, y)$, a basic set of dyadic relations are defined: '$DC(x, y)$' ('x is disconnected from y'), '$P(x, y)$' ('x is a part of y'), '$PP(x, y)$' ('x is a proper part of y'), '$x = y$' ('x is identical with y'), '$O(x, y)$' ('x overlaps y'), '$DR(x, y)$' ('x is discrete from y') '$PO(x, y)$' ('x partially overlaps y'), '$EC(x, y)$' ('x is externally connected with y)', '$TPP(x, y)$' ('x is a tangential proper part of y') and '$NTPP(x, y)$' ('x is a nontangential proper part of y'). The relations: P,PP,TPP and NTPP being non-symmetrical support inverses. For the inverses we use the notation Φ^{-1}, where $\Phi \in$ {P,PP,TPP and NTPP}. Of the defined relations, DC,EC,PO,=,TPP,NTPP and the inverses for TPP and NTPP are provably mutually exhaustive and pairwise disjoint. The complete set of relations described above can be embedded in a relational lattice. This is given in Figure 1. The symbol $\top$ is interpreted as tautology and the symbol $\bot$ as contradiction. The ordering of these relations is one of subsumption with the weakest (most general) relations connected directly to top and the strongest (most specific) to bottom. For example, TPP implies PP, and PP implies either TPP or NTPP. A greatest lower bound of bottom indicates that the relations are mutually disjoint, for example with TPP and NTPP, and P and DR. This lattice corresponds to a set of theorems (eg. $\forall x y[PP(x, y) \rightarrow [TPP(x, y) \vee NTPP(x, y)]]$) which we have verified.

$DC(x, y) \equiv_{def} \neg C(x, y)$
$P(x, y) \equiv_{def} \forall z[C(z, x) \rightarrow C(z, y)]$
$PP(x, y) \equiv_{def} P(x, y) \wedge \neg P(y, x)$
$x = y \equiv_{def} P(x, y) \wedge P(y, x)$
$O(x, y) \equiv_{def} \exists z[P(z, x) \wedge P(z, y)]$
$PO(x, y) \equiv_{def} O(x, y) \wedge \neg P(x, y) \wedge \neg P(y, x)$
$DR(x, y) \equiv_{def} \neg O(x, y)$
$TPP(x, y) \equiv_{def} PP(x, y) \wedge \exists z[EC(z, x) \wedge EC(z, y)]$
$EC(x, y) \equiv_{def} C(x, y) \wedge \neg O(x, y)$
$NTPP(x, y) \equiv_{def} PP(x, y) \wedge \neg \exists z[EC(z, x) \wedge EC(z, y)]$
$P^{-1}(x, y) \equiv_{def} P(y, x)$
$PP^{-1}(x, y) \equiv_{def} PP(y, x)$
$TPP^{-1}(x, y) \equiv_{def} TPP(y, x)$
$NTPP^{-1}(x, y) \equiv_{def} NTPP(y, x)$

In the original theory, several other defined relations (missing here) were defined. These were the set of relations: '$TP(x, y)$' ('x is a tangential part of y'), '$NTP(x, y)$' ('x is a nontangential part of y'),'$TPI(x, y)$' ('x is the identity tangential part of y'), and, '$NTPI(x, y)$' ('x is the identity nontangential part of y'). We also omit in the new theory the set of topological functions introduced by Clarke, and adopted by us in the original theory. In this revised theory, we make no formal distinction in our model between open, semi-open and closed regions used to interpret this part of the formalism (as was done in the original theory), so for example now the identity relation does not split into two specialisations here, as it did in the original theory to account for the differences between types of regions. A similar rationale applies for the explicit introduction of the tangential part and nontan-

[5] In Clarke's theory and in our original theory, when two regions x and y connect, they are said to share a point in common; thus the interpretation of the connects relation in the new theory is weaker.

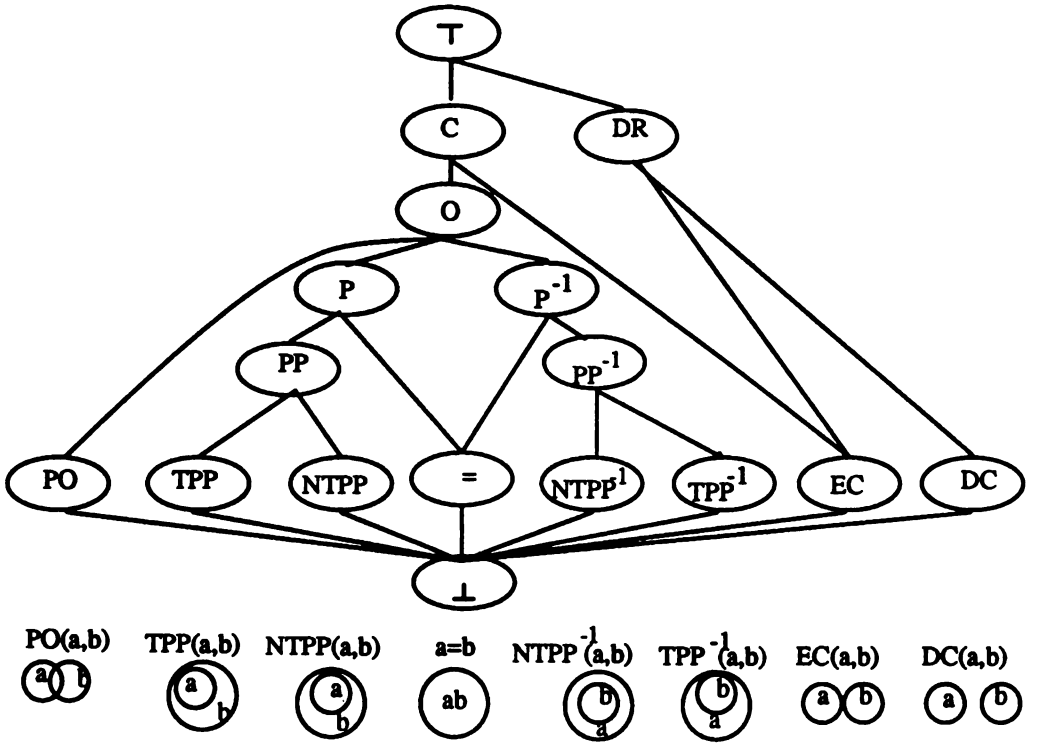


Figure 1: A lattice defining the subsumption hierarchy of the dyadic relations defined solely in terms of the primitive relation $C(x, y)$.

gential part relations mentioned above - see Randell (1991), Randell and Cohn (1989), Randell and Cohn (1992) for further details.

Excepting the definition for the complement of a region, the Boolean part of the new theory follows the original theory, and Clarke's. The Boolean functions[6] are: '$\mathrm{sum}(x, y)$' which is read as 'the sum of x and y', 'Us' as 'the universal (spatial) region', '$\mathrm{compl}(x)$' as 'the complement of x', '$\mathrm{prod}(x, y)$' as 'the product (i.e. the intersection of x and y' and '$\mathrm{diff}(x, y)$' as 'the difference of x and y'. The functions: '$\mathrm{compl}(x)$', '$\mathrm{prod}(x, y)$' and '$\mathrm{diff}(x, y)$' are partial but are made total in the sorted logic by simply specifying sorts restrictions and by introducing a new sort called NULL. The sorts NULL and REGION are disjoint.

$$\mathrm{sum}(x, y) =_{def} \iota y[\forall z[\mathrm{C}(z, y) \leftrightarrow [\mathrm{C}(z, x) \vee \mathrm{C}(z, y)]]]$$
$$\mathrm{compl}(x) =_{def} \iota y[\forall z[[\mathrm{C}(z, y) \leftrightarrow \neg \mathrm{NTPP}(z, x)] \wedge [\mathrm{O}(z, y) \leftrightarrow \neg \mathrm{P}(z, x)]]]$$
$$\mathrm{Us} =_{def} \iota y[\forall z[\mathrm{C}(z, y)]]$$
$$\mathrm{prod}(x, y) =_{def} \iota z[\forall u[\mathrm{C}(u, z) \leftrightarrow \exists v[\mathrm{P}(v, x) \wedge \mathrm{P}(v, y) \wedge \mathrm{C}(u, v)]]]$$
$$\mathrm{diff}(x, y) =_{def} \iota w[\forall z[\mathrm{C}(z, w) \leftrightarrow \mathrm{C}(z, \mathrm{prod}(x, \mathrm{compl}(y)))]]$$
$$\forall xy[\mathrm{NULL}(\mathrm{prod}(x, y)) \leftrightarrow \mathrm{DR}(x, y)]$$

In Clarke (1981, 1985) (and also in our original theory) the complement definition is defined so that a region y connects with the complement of region x if and only if y is not a part of x. This has the formal consequence that no region is connected with its own complement.[7] However, this result is not formally derivable in the new theory, and moreover must not be so given the new interpretation. This arises from the new interpretation for the connects relation, since every region (which is

[6] $\alpha(\overline{x}) =_{def} \iota y[\Phi[\alpha(\overline{y})]$ means $\forall \overline{x}[\Phi(\alpha(\overline{x}))]]$; thus, e.g., the definition for $\mathrm{prod}(x, y)$ is translated out (in the object language) as: $\forall xyz[\mathrm{C}(z, \mathrm{prod}(x, y)) \leftrightarrow \exists w[\mathrm{P}(w, x) \wedge \mathrm{P}(w, y) \wedge \mathrm{C}(z, w)]]$.

[7] Here we are assuming certain restrictions on x. In the unsorted theory assumed by Clarke, this amounts to x not being identical to the universal region - our constant Us.

not identical to the universal region) will be connected with its own complement. In fact this difference is reflected in the theorem: $\forall x \mathrm{EC}(x, \mathrm{compl}(x))$ which contradicts the related theorem described above.

An additional axiom is then added to the new theory which stipulates that every region has a nontangential proper part:

$$\forall x \exists y[\mathrm{NTPP}(y, x)] \qquad \text{(i)}$$

This axiom mirrors a formal property of Clarkes' theory, where he stipulates that every region has a nontangential part, and thus an interior (remembering that in Clarke's theory a topological interpretation is assumed).

4.1 One piece regions

Clarke's theory supports a model where regions may topologically connected (i.e. in one piece) or disconnected (in more than one piece). A definition for a connected region is given - which states that a region is disconnected iff it cannot be split into two disjoint parts. The same type of model supporting either individual connected or disconnected regions appears in the new theory, only here the definition for an individual connected region does not need to incorporate the distinction between topological types of regions, i.e. being open, semi-open or closed. The definition simply states that an individual region is connected if it cannot be split into parts whose union is that region, and where these parts are not connected to each other[8], i.e.

$$\mathrm{CON}(x) \equiv_{def} \forall yz[\mathrm{sum}(y, z) = x \rightarrow \mathrm{C}(y, z)]$$

4.2 Proper and Improper regions

A proper region is defined to be a region that has a nontangential proper part, and an improper region, a region that is not a proper region. In the basic theory, where we allow space to be continuously decomposed into a set of nontangential proper parts, every region becomes a proper region, and no region an improper region. However, in section 5 we discuss the possibility of adding atoms into the formal theory, and by positing atoms, improper regions can be defined. Examples of improper regions would be single atoms, and various clusters of atoms forming strings, rings, and sheets in 3-space. As the possibility of defining these objects requires atoms to be posited, and that the question of whether or not atoms can be included is a complex one, we refer the reader to section 5 where this matter is discussed in more detail.

[8]The original definition in Randell and Cohn (1989) had to be modified since it referred to the closure of a region.

4.3 Inclusion vs Containment

As with the original theory (but missing in Clarke) a primitive function 'conv(x)' ('the convex-hull of x') is defined and axiomatised. We assume here that conv is only well sorted when defined on one piece regions.

$$\forall x \mathrm{P}(x, \mathrm{conv}(x))$$
$$\forall x \mathrm{P}(\mathrm{conv}(\mathrm{conv}(x)), \mathrm{conv}(x))$$
$$\forall x \forall y \forall z[[\mathrm{P}(x, \mathrm{conv}(y) \wedge \mathrm{P}(y, \mathrm{conv}(z))] \rightarrow \mathrm{P}(x, \mathrm{conv}(z))]$$
$$\forall x \forall y[[\mathrm{P}(x, \mathrm{conv}(y)) \wedge \mathrm{P}(y, \mathrm{conv}(x))] \rightarrow \mathrm{O}(x, y)]$$
$$\forall x \forall y[[\mathrm{DR}(x, \mathrm{conv}(y)) \wedge \mathrm{DR}(y, \mathrm{conv}(x))] \leftrightarrow \mathrm{DR}(\mathrm{conv}(x), \mathrm{conv}(y))]$$

We use this function to define a set of relations which describe regions being inside, partially inside and outside, e.g. 'INSIDE(x, y)' (x is inside y'), 'P-INSIDE(x, y)' ('x is partially inside y') and 'OUTSIDE(x, y)' ('x is outside y'). This particular set of relations extends below DR(x, y) in the basic theory. The developed theory actually supports many specialisations of these particular relations, with, for example, one region being wholly outside, or just outside, or just inside, or wholly inside another - see Randell and Cohn (1989, 1992) and Randell (1991). However, here we restrict the set of defined relations to the specialisations given above, their inverses, and the set of relations that result from non-empty intersections. The set of base relations for this particular set are finally generated by defining a further set of specialisations of these relations using the EC and DC relations. In the interest of space, only a subset of the constructible defined relations are given below. However the interested reader should have no difficulties actually generating the formal definitions from the schema given below.

Here are the formal definitions for the named relations introduced above, together with their inverses:

$$\mathrm{INSIDE}(x, y) \equiv_{def} \mathrm{DR}(x, y) \wedge \mathrm{P}(x, \mathrm{conv}(y))$$
$$\text{P-INSIDE}(x, y) \equiv_{def} \mathrm{DR}(x, y) \wedge \mathrm{PO}(x, \mathrm{conv}(y))$$
$$\mathrm{OUTSIDE}(x, y) \equiv_{def} \mathrm{DR}(x, \mathrm{conv}(y))$$
$$\mathrm{INSIDE}^{-1}(x, y) \equiv_{def} \mathrm{INSIDE}(y, x)$$
$$\text{P-INSIDE}^{-1}(x, y) \equiv_{def} \text{P-INSIDE}(y, x)$$
$$\mathrm{OUTSIDE}^{-1}(x, y) \equiv_{def} \mathrm{OUTSIDE}(y, x)$$

A new set of base relations (using the relations defined immediately above) are constructed according to the following schema:

$$\alpha\_\beta(x, y) \equiv_{def} \alpha(x, y) \wedge \beta(x, y)$$

where: $\alpha \in$ {INSIDE, P-INSIDE, OUTSIDE}, and $\beta \in$ {INSIDE^{-1}, P-INSIDE^{-1}, OUTSIDE^{-1}}, excepting where α = INSIDE and β =INSIDE^{-1} Each of these 'composite' relations then split into two variants, the case where x and y EC, and the case where they DC. This finally gives rise to the new set of base relations in the extended theory, which now number 22

instead of 8 in the revised basic theory (cf. 23 and 9 in the original theory).

Two functions capturing the concept of the inside and the outside of a particular region are also definable (where 'inside(x)' is read as 'the inside of x', and 'outside(x)' as 'the outside of x' respectively:

$$\text{inside}(x) =_{def} \iota y[\forall z[\text{C}(z,y) \leftrightarrow \exists w[\text{INSIDE}(w,x) \wedge \text{C}(z,w)]]]$$
$$\text{outside}(x) =_{def} \iota y[\forall z[\text{C}(z,y) \leftrightarrow \exists w[\text{OUTSIDE}(w,x) \wedge \text{C}(z,w)]]]$$

4.4 Geometrically Inside vs Topologically Inside

In the previous section the DR relation is specialised to cover relations describing objects being either inside, partially inside or outside other objects. However this ignores some useful distinctions that can be drawn between different cases of bodies being inside another. In this case we separate out the case where one body is topologically inside another, and where one body is inside another but not topologically inside – this we call being geometrically inside. The important point of one body being topologically inside another is that one has to 'cut' through the surrounding body in order to reach and make contact with the contained body. In the geometrical variant this is not the case.

Figure 2: The distinction between being topologically and geometrically inside. The dotted lines appearing here and in Figure 3 indicate the extent of the convex hull of the surrounding bodies.

$$\text{TOP-INSIDE}(x,y) \equiv_{def} \text{INSIDE}(x,y) \wedge \forall z[[\text{CON}(z) \wedge \text{C}(z,x) \wedge \text{C}(z,\text{outside}(y))] \rightarrow \text{O}(z,y)]$$
$$\text{GEO-INSIDE}(x,y) \equiv_{def} \text{INSIDE}(x,y) \wedge \neg\text{TOP-INSIDE}(x,y)$$

It is also possible to specialise the relation of being geometrically inside – in this case setting up definitions to distinguish between the following pictorial representations – Figure 4:

In order to make this formal distinction we first set up a stronger case of a connected or one-piece region to that assumed above. The important part of the following definition is the $\text{P}(\text{conv}(\text{sum}(v,w)),x)$ literal in the consequent of the definiens. This condition ensures that the connection between any two parts of a region whose sum equals that region, is not point or edge

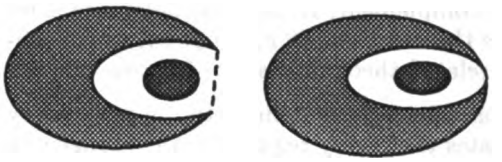


Figure 3: Two variants of being geometrically inside. In the right hand figure the two 'arms' meet at a point.

connected. That is to say it ensures a 'channel' region exists connecting any two connected parts. This notion of being connected mirrors and simplifies our previous definition of a quasi-manifold – in this case we use the concept of a convex body rather than use topological and Boolean concepts in the earlier definition – see Randell and Cohn (1989).

$$\text{CON}'(x) \equiv_{def} \text{CON}(x) \wedge \forall yz[[\text{sum}(y,z) = x \rightarrow \text{C}(y,z)] \rightarrow \exists vw[\text{P}(v,y) \wedge \text{P}(w,z) \wedge \text{P}(\text{conv}(\text{sum}(v,w)),x)]]$$

Now we give the formal distinction between the two cases of being geometrical inside. In the first case a 'channel' region exists connecting the outside of the surrounding body with the contained body, in the second case the surrounding body has closed forming (in this case) a point connection. In both cases we can see how in contrast with the notion of being topologically inside, it is possible to construct a line segment that connects with both the surrounding body and the contained body without cutting through the surrounding body. Definitions distinguishing between the two cases are as follows, where the open and closed variants respectively refer to the first and second cases described above.

$$\text{GEO-INSIDE-OPEN}(x,y) \equiv_{def} \text{GEO-INSIDE}(x,y) \wedge \text{CON}'(\text{sum}(\text{inside}(y),\text{outside}(y)))$$
$$\text{GEO-INSIDE-CLOSED}(x,y) \equiv_{def} \text{GEO-INSIDE}(x,y) \wedge \text{CON}(\text{sum}(\text{inside}(y),\text{outside}(y))) \wedge \neg\text{CON}'(\text{sum}(\text{inside}(y),\text{outside}(y)))$$

4.5 Theorems in the new theory

As mentioned above, some important differences exist between both Clarke's and the original theory, and the new theory. For brevity we shall subsume our original theory under Clarke's, when making the contrast. Where a difference arises between some theorem of Clarke's and our own original theory, we shall make this explicit. First we demonstrate how the topological distinction drawn between open, semi-open and closed regions sanctioned in Clarke's theory cannot be made in the new theory. For Clarke, two regions x and y are identical iff any region connecting with x connects with y and vice-versa, i.e. $\forall xy[x = y \leftrightarrow$

$\forall z[\mathrm{C}(z,x) \leftrightarrow \mathrm{C}(z,y)]]$; however in the new theory, an additional theorem concerning identity becomes provable which is not a theorem in Clarke's theory. This is: $\forall xy[x = y \leftrightarrow \forall z[\mathrm{O}(z,x) \leftrightarrow \mathrm{O}(z,y)]]$. The topological model used in Clarke's theory, together with the absence of boundary elements as regions, explains why this formula is not derivable. For example, given the closure of region x and its interior, then any region overlapping the closure of x, overlaps the interior of x, and vice-versa, (remembering that overlapping regions entail that they share a common interior point) but from this we cannot allow the interior of x to be identical with its closure, which would follow if the related formula were to be a theorem in Clarke's theory.

The next important difference between Clarke's and the new theory is the formula: $\forall xy[\mathrm{PP}(x,y) \rightarrow \exists z[\mathrm{P}(z,y) \wedge \neg\mathrm{O}(z,x)]]$ which is provable in the new theory, but not in Clarke's. Given Clarke's theory supports open, semi-open and closed regions as a model, it becomes clear why this formula is not provable in Clarke's theory, since while the interior of a region is a proper part of its closure, (and boundaries are not regions) there is no other part of the closure which does not overlap the interior. If one adds the condition that the regions in question are closed, then, the formula is true of Clarke's theory, but this condition is waivered in the new theory. Another related formula is: $\forall xy[\mathrm{PO}(x,y) \rightarrow [\exists z[\mathrm{P}(z,y) \wedge \neg\mathrm{O}(z,x)] \wedge \exists w[\mathrm{P}(w,x) \wedge \neg\mathrm{O}(w,y)]]]$, which is a theorem in the new theory but not in Clarke's. A counter example arises in Clarke's theory where we have two semi-open spherical regions, x and y (with identical radii), such that the northern hemisphere of x is open and the southern hemisphere is closed, and the northern hemisphere of y is closed and the southern hemisphere open. If x and y are superimposed so that their centres and equators coincide, then x and y will partially overlap, but no part of x is discrete from y, and vice-versa. Both these theorems in the new theory show that a positive Boolean difference exists between y and x when x is a proper part of y. Again in Clarke's theory this result only follows when both x and y are closed regions.

In the new theory, $\forall x\ \mathrm{EC}(x, \mathrm{compl}(x))$ holds; this contrasts with the theorem: $\forall x\ \mathrm{DC}(x, \mathrm{compl}(x))$ in both the original and in Clarkes' theories. Also here it is worth pointing out that in the original theory (which included Clarke's set of topological operators) we included the axiom: $\forall x\mathrm{EC}(\mathrm{cl}(x), \mathrm{cl}(\mathrm{compl}(x)))$ which ensured that the closure of x externally connected with the closure of the complement of x, where x was restricted so that it was not the universal region.[9]

Other interesting theorems are: $\forall xyz[[\mathrm{C}(z,y) \wedge \neg\mathrm{C}(z,x)] \rightarrow \exists w[\mathrm{P}(w,y) \wedge \neg\mathrm{O}(w,x) \wedge \mathrm{C}(z,w)]]$, and $\forall xy[[\mathrm{PP}(x,y) \wedge \mathrm{Connected}(y)] \rightarrow \exists z[\mathrm{P}(z,y) \wedge \mathrm{EC}(z,x)]]$. Note for the latter formula to be a theorem, an additional restriction on variable y is required, namely that y is a place-holder for a one-piece region.

Readers familiar with either Clarke's theory, or our own original theory may be wondering what happens to the relations TP and NTP which are excluded here. In the new theory, we find that if we defined these relations and added them to the extant set, the two relations would give rise (on the assumption that x is not identical with the universal spatial region) to the theorems: $\forall x\mathrm{TP}(x,x)$ and $\forall x\neg\mathrm{NTP}(x,x)$ respectively. The latter indicates that no positive instance of the relation NTP which is not a case of NTPP can arise in any model of the new theory. Thus we omit NTP and TP for reasons of symmetry and neither relation appears in the relational lattice depicted in Figure 1.

4.6 Transitivity Tables for the new theory

A transitivity table is defined as follows. Given a particular theory Σ supporting a set of mutually exhaustive and pairwise disjoint dyadic relations, three individuals, a, b and c and a pair of dyadic relations R_1 and R_2 selected from Σ such that $R_1(a,b)$ and $R_2(b,c)$, the transitive closure $R_3(a,c)$ represents a disjun ction of all the possible dyadic relations holding between a and c in Σ. Each $R_3(a,c)$ result can b e represented as one entry of a matrix for each $R_1(a,b)$ and $R_2(b,c)$ ordered pair. If there are n dyadic relations supported by Σ, then there will be $n \times n$ entries in the matrix. T his matrix is called a transitivity table. A well known example of a transitivity table appears in an implementation of Allen's temporal logic (Allen 1983); we also give a transitivity table for our original theory in (Randell, Cohn and Cui 1992) and in (Randell and Cohn 1992).

The new transitivity table is essentially the same as the original one excepting that the new matrix for the basic set of base relations has only one relation covering the identity relation, and not two as before. The new table is easily constructed by simply eliminating the row and column labelled NTPI, eliminating every NTPI entry which appears in each cell, and replacing TPI with =. The table is an 8×8 matrix (64 cells) averaging ≈ 3 entries per cell. For the basic extension to this table (including the inside, partially inside and outside relations) the matrix increases to 22×22 (484 cells) averaging ≈ 9 entries per cell. On these two examples, the increase in the number of base relations does not appear to increase the complexity of the number of entries in the cells generated. The extended transitivity table furhter increases to 30×30 (900 cells) with the specialisation of the inside relation covering the distinction between being topologically inside and being geometrically inside. Note that this does not exhaust the maximal number of base rela-

[9] It turns out that if we assume the universal region is topologically connected (which is a definable concept in Clarkes' theory) and that x is not the universal region, we can prove this as a theorem. We are indebted to Laure Vieu who demonstrated the proof to us.

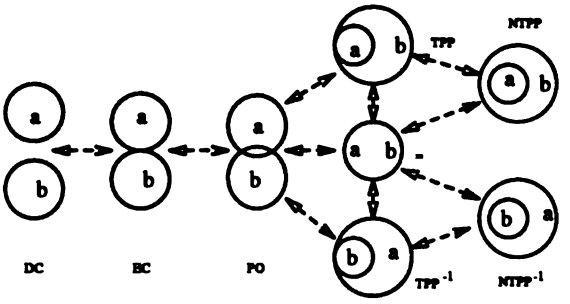


Figure 4: A pictorial representation of the base relations and their direct topological transitions.

tions that can be generated from the definitions given in this paper – for we have not taken into account the distinction made between the two defined cases of one region being geometrically inside another.

As each cell in a transitivity table corresponds to a theorem, computing these large transitivity tables is a non-trivial task – see Randell, Cohn and Cui (1992). We have recently simplified this task by using a program that uses a bit-string model to generate all possible transitivity table configurations for a given set of base relations. The original 9 × 9 table has been formally proved, and the program constructed to generate the larger 22 × 22 table conforms with the predicted entries for the 8 × 8 table.[10]

4.7 Envisioning axioms in the new theory

As mentioned above, we express different sets of base relations in the form of a set of envisioning axioms. These stipulate direct transitions that are allowed between pairs of objects over time. A pictorial representation of the basic set of base relations and their direct topological transitions in the new theory is given in Figure 4.[11]

For the basic set of base relations - the set DC,EC,PO,=, TPP,NTPP and the inverses for TPP and NTPP, no practical difference arises from that used in the original theory (using the set of 9 base relations). This arises simply because in the domains we modelled we simply mapped the named individuals to closed regions, thus eliminating the base relation NTPI which is only true for open regions. However, if we add NTPI into the envisioning axioms, then the number of paths connecting nodes in the graph for the 9 base relations compared with that for the graph generated from the new theory (with 8 base relations) reduces from 17 to 11.[12]

4.8 Models and structures for the new theory

We have already given one model for the new theory, interpreting the C relation in terms of two regions whose closures share a common point. However, other models exist. We could simply state that two regions connect, when it is not possible to 'fit' another distinct region between the two, or alternatively to say when the distance between them is zero. Clarke (1981) only suggests the point based topological interpretation as one possible interpretation for his axiomatisation.

For the new theory there is an important meta-theoretic restriction concerning Boolean sums or unions of regions, namely that infinite unions cannot be allowed. If infinite unions are allowed, the theory becomes inconsistent. The proof sketch is as follows. In the theory we have an axiom that ensures every region has a nontangential proper part, and a theorem that states that if region x is a proper part of region y, then there exists another region z that is part of y, but is disjoint with x. From both of these, it follows that every region is subdivided into an infinite set of nontangential proper parts. However if we take the infinite union of all the nontangential proper parts of y, then in the limit this union becomes identical with y. However, the definition for NTPP requires no region to externally connect with y, but y now identical with the infinite union of all its nontangential proper parts, must externally connect with its own complement – which is inconsistent. Viewed another way this result simply illustrates the fact that (on pain of contradiction) interiors (in the topological sense of the term) cannot be explicitly introduced into the theory. In fact it can be shown that by adding the definition (for the interior of region x):

$$\mathrm{int}(x) =_{def} \iota y[\forall z[\mathrm{C}(z,y) \leftrightarrow \exists w[\mathrm{NTPP}(w,x) \wedge \mathrm{C}(z,w)]]$$

and positing the existence of interiors, a formal contradiction is generated.

4.9 Comparisons with the Classical Calculus of Individuals

Readers familiar with Leonard and Goodman's (1940) (classical) calculus of individuals will notice similar-

[10]The generation of the 22 × 22 table took 2 days CPU time on a Sun Sparc IPC.

[11]Note that here as in the network used in the original theory, we have assumed that the regions depicted have nontangential proper parts. This means no direct transition from e.g. EC to identity is allowed which would arise if both regions were atomic.

[12]This assumes that a legitimate path connects the node NTPI with the nodes TPI,PO,TPP,NTPP and the inverses for TPP and NTPP. Ontologically speaking this is the most liberal result where we allow regions to change their topological type over time, i.e. from non-open to open as in for example the path linking TPI and NTPI. Other less liberal linkages may well be envisaged which would reduce the number of connections between nodes.

ities between this calculus and the new calculus described above. In the classical calculus, DR is axiomatised to be irreflexive and symmetrical, and is used to create a set of dyadic relations and Boolean operators defined on individuals. No analogues of DC and EC (defined in Clarke's calculus) are defined in the classical calculus. With the weaker relation C this distinction can be made. The new theory contains, as part of its complement definition, a conjunct that mirrors the definition for complementation in the classical calculus, i.e. the formula: $\forall xy[O(x, compl(y)) \leftrightarrow \neg P(x, y)]$. This conjunct forces the following formula: $\forall xy[P(x, y) \leftrightarrow \forall z[O(z, x) \rightarrow O(z, y)]$ to be a theorem in the new theory; in fact this equivalence mirrors the definition for P in the classical calculus, where P is defined solely in terms of O. The new theory straddles between Clarke's and the classical calculi of individuals.

5 Atomic regions: a discussion

In Randell, Cui and Cohn (1992) we allowed atomic regions or atoms to be introduced into the ontology. Atoms were defined as regions with no proper parts, and an existential axiom was added that ensured every region had an atom as a part. In the intended model, atoms were understood to be 'very small' regions. Atoms were then used in the definition of what we called the skin of a region. This skin is comparable to the notion of a mathematical surface, except that unlike a surface proper, the skin of a region was understood to have non-zero thickness. The definition for the skin of a region simplified the analogous definition given in Randell (1991), and in Randell and Cohn (1992); and was a direct result of our new theory.

As the basic theory supports a model with a continuous decomposition of regions into nontangential proper parts (being a direct consequence of axiom (i) above), some restriction was necessary to avoid building inconsistency into the theory. Axiom (ii) used below consequently replaced axiom (i). The basic extension was presented as follows:

$$\begin{array}{l} ATOM(x) \equiv_{def} \forall y[P(y, x) \rightarrow y = x] \\ \forall x[\neg ATOM(x) \rightarrow \exists y[ATOM(y) \wedge P(y, x)]] \\ \forall x[\neg ATOM(x) \rightarrow \exists y[NTPP(y, x)]] \qquad \text{(ii)} \\ skin(x) =_{def} \iota y[\forall z[C(z, y) \leftrightarrow \exists v[ATOM(v) \wedge \\ \qquad\qquad TPP(v, x) \wedge C(z, v)]]] \end{array}$$

5.1 Small is not beautiful: problems posed by atoms

Unfortunately, it turns out that even given the restriction imposed by axiom (ii) above, this is not sufficient to stop inconsistency arising in the atomic variant of this theory. The discovery of this came as something of a surprise, since the axioms and definitions used seemed intuitively correct, until the discovery of the contradiction forced us to look deeper into the axiomatisation. The proof is as follows. Assume an arbitrary atom, call it b. Assume b is not identical to the universe, then b has a complement, and EC(b,compl(b)) follows. However, since b is an atom, then everything connected to b, connects with compl(b). From the definition of P, P(b,compl(b)) also follows. P(b,compl(b)) implies O(b,compl(b)), which implies ¬EC(b,compl(b)). Thus EC(b,compl(b)) and ¬EC(b,compl(b)) – contradiction – QED. The problem lies with the definition of P, which (in the intended domain) is false for atoms, for it is not true that just because every region connected to an atom is connected to its complement, that atom is necessarily part of its complement.

Now it turns out that Clarke's theory is immune from this problem for the following reasons: (i) his definition for complement (being different) ensures that -C(x,compl(x)) follows, and (ii) because of the existence of interiors in his theory. This latter feature ensures the existence of some region that (connects with itself but) does not connect with its complement. And thus it does not follow that an atom posited in Clarke's theory (being identical to its own interior) is part of its own complement. The explicit introduction of (topological) interiors into Clarke's calculus ensures his theory is (at least on this point) sound, but given we sought to eliminate this explicit characterisation of open, semi-open and closed regions other solutions must be sought.

5.2 Small is not beautiful: solutions

Below we give three potential solutions to the problem posed by admitting atoms into the domain. Two of these require that atoms be introduced as a primitive sort, while the third keeps atoms as a definable sort but introduces points. This section covers work in progress, so the proposed solutions must viewed in this light.

The first solution is to make atoms a primitive sort, that is to say we do not give a formal definition for atom as above. Atoms are then allowed to have regions as parts which we call particles, but with the restriction that particles always occur in atoms and do not appear in non-atomic regions without also being embedded in atoms. Thus what we call atoms here are really pseudo-atoms since they contain proper parts. The idea is that for most practical modelling purposes (pseudo) atoms are considered to be the most primitive entity that is explicitly referred to. The contradiction arising from positing atoms now dissolves. Because atoms now have proper parts, this means that it is no longer true that every region which connects with an atom connects with its complement.

Given atoms are represented as a primitive sort, we need to axiomatise their properties. First we stipu-

late that any two atoms that overlap become identical. Next we add the definition for a particle, together with an axiom that every atom has a particle as a proper part:

$$\forall xy[[\mathrm{ATOM}(x) \wedge \mathrm{ATOM}(y) \wedge \mathrm{O}(x,y)] \rightarrow x = y]$$
$$\mathrm{PARTICLE}(x) \equiv_{def} \exists y[\mathrm{ATOM}(y) \wedge \mathrm{PP}(x,y)]$$
$$\forall x[[\mathrm{ATOM}(x) \rightarrow \exists y[\mathrm{PARTICLE}(y) \wedge \mathrm{PP}(y,x)]$$
$$\forall x[[\neg\mathrm{ATOM}(x) \wedge \neg\mathrm{PARTICLE}(x)] \rightarrow \exists y[\mathrm{P}(y,x) \wedge \mathrm{ATOM}(y)]]$$

The second solution takes atoms and the summation operator as a primitive sort and function respectively and then defines the part/whole relation in terms of summed regions. First we axiomatise C as before and define C on atoms. Then we define the relations DC, = and EC for atoms. (Here it is useful to remember that atoms can only be disconnected, externally connected or be identical.) Axioms defining the standard properties of the summation operator are then given, together with an axiom that ensures that if two atoms are disjoint, their sum is not an atom:

$$\mathrm{DC}(x,y) \equiv_{def} \neg\mathrm{C}(x,y)$$
$$x = y \equiv_{def} \forall z[\mathrm{C}(z,x) \leftrightarrow \mathrm{C}(z,y)]$$
$$\mathrm{EC}(x,y) \equiv_{def} \mathrm{C}(x,y) \wedge \neg(x = y)$$

$$\forall x \mathrm{sum}(x,x) = x$$
$$\forall xy[\mathrm{sum}(x,y) \rightarrow \mathrm{sum}(y,x)]$$
$$\forall xyz[\mathrm{sum}(x,\mathrm{sum}(y,z)) = \mathrm{sum}(\mathrm{sum}(x,y),z)]$$
$$\forall xy[\neg(x = y) \rightarrow \neg\mathrm{ATOM}(\mathrm{sum}(x,y))]$$
$$\mathrm{REGION}(x) \equiv_{def} \forall y[\mathrm{C}(y,x) \leftrightarrow \exists z[\mathrm{ATOM}(z) \wedge \mathrm{P}(z,x) \wedge \mathrm{C}(y,z)]]$$

Note that for the first group of formulae presented immediately above, all the variables are of sort ATOM; this restriction is relaxed in the second group where all the variables are of sort SPATIAL (remembering that ATOM is a subsort of SPATIAL in this theory).

Next we start to define the set of binary relations which are true for non-atomic regions:

$$\mathrm{P}(x,y) \equiv_{def} \exists z[y = \mathrm{sum}(x,z)]$$
$$\mathrm{O}(x,y) \equiv_{def} \exists z[\mathrm{P}(z,x) \wedge \mathrm{P}(z,y)]$$
$$\mathrm{EC}(x,y) \equiv_{def} \neg\mathrm{O}(x,y) \wedge \exists zu[\mathrm{ATOM}(z) \wedge \mathrm{ATOM}(u) \wedge \mathrm{P}(z,x) \wedge \mathrm{P}(u,y) \wedge \mathrm{EC}(z,u)]$$

The reader should now be able to complete the set of binary relations defined on non-atomic regions, using the earlier set of definitions as a guide. The rest of the axiomatisation then follows as before, excepting of course that the summation operator does not now appear as a definition.

The third solution keeps atoms as a defined sort, but also introduces points as a new primitive sort into the ontology. The general idea is to rework the definition of the part/whole relation in terms of points instead of regions and connection as before.

First the new sort POINT is stipulated to be pairwise disjoint with REGION and NULL. A new primitive relation 'IN(x,y)' read as '(point) x is incident in (region) y' is then added; this replaces the primitive C relation used above. IN is axiomatised to be irreflexive and asymmetrical[13]. Then we define both the C and P relation in terms of points, instead of regions as before:

$$\forall x \neg\mathrm{IN}(x,x)$$
$$\forall xy[\mathrm{IN}(x,y) \rightarrow \neg\mathrm{IN}(y,x)]$$
$$\mathrm{C}(x,y) \equiv_{def} \exists z[\mathrm{IN}(z,x) \wedge \mathrm{IN}(z,y)]$$
$$\mathrm{P}(x,y) \equiv_{def} \forall z[\mathrm{IN}(z,x) \rightarrow \mathrm{IN}(z,y)]$$

The crucial point(!) here is that the formula: $\forall xy[\mathrm{P}(x,y) \leftrightarrow \forall z[\mathrm{C}(z,x) \rightarrow \mathrm{C}(z,y)]]$ is now not provable; this serves to block the proof which generated the contradiction described above.

The rest of the axiomatisation then follows that given in the main body of this paper. Note here that we have chosen to replace C with IN as the primitive dyadic relation upon which this axiomatisation is built. It is certainly possible to axiomatise C as before and then axiomatise IN in terms of C and P, i.e. stipulating that two regions connect iff they share a common incident point, and stipulating that one region is part of another iff every point incident in the former is incident in the latter. Our choice is simply based on ontological parsimony, for while connection can be defined in terms of incidence, incidence cannot be defined in terms of connection.

6 Related and Further Work

We have already mentioned Clarke's calculus of individuals, our earlier work of which this present theory is a simplification, and Allen's and Hamblin's work on interval logics. The only other work of which we are aware, that uses Clarke's theory for describing space, is Aurnague (1991) and Vieu (1990). Other work on the description of space using a body rather than a point based ontology, can be found in Laguna (1922), Tarski(1956) and Whitehead (1978). There have been some attempts in the qualitative spatial reasoning literature to employ Allen's interval logic, for describing space, see for example Freksa (1990) and Hernandez (1990), but here a stronger primitive relation used, which does not allow the full range of topological relationships to be formally described as given in both Clarkes' and our original and new theories. Apart from the question raised by adding atoms to the theory, we are currently working on the question as to whether the new theory supports decidable subsets. We have

[13] Note that the two axioms for IN are not required in a sorted logic.

already indicated some extensions to this new logic above, including a temporal extension and extending the ontology further to be able to reason about bodies and describe, states, events and processes. For other extensions to the spatial theory itself, work described in Randell (1991) can also be included. For example, we could add a metric extension to the theory, using either a distance function, or alternatively by adding a ternary relation (along the lines of Van Benthem 1982, appendix A) that gives comparative distances between objects.

7 Conclusions

The new theory gains over Clarke's theory and the original theory we developed from several viewpoints: ontologically (the explicit distinction between open, semi-open and closed regions is eliminated), definitional (there are fewer defined predicates, and fewer axioms), metatheoretically (there are fewer entries in transitivity tables, and fewer nodes in the the sort hierarchy), and computationally (for comparable theorems, there are fewer formulae in the search space, and fewer nested functions to address in definitions).

The major difference at first sight is ontological parsimony, but we argue that the loss of granularity is not important when modelling physical domains, since physical objects correspond to 'closed' regions, and boundaries can be modelled using 'skins'.

Acknowledgements

This research is supported by SERC under grant no. GR/G64388 which we gratefully acknowledge.

References

[1] J. F. Allen: " Maintaining Knowledge about Temporal Intervals," Comm. ACM26(11), 1983

[2] J. F. Allen and P. J. Hayes: "A Common Sense Theory of Time," in *Proc. IJCAI*, Morgan Kaufmann, Los Altos., 1985

[3] J. F. Allen and P. J. Hayes: "Moments and Points in an Interval-Based Temporal Logic," (TR180), Rochester, NY, Depts of Comp. Sci. and Phil., 1987

[4] J. F. Allen and P. J. Hayes: "Short time periods," *Proc IJCAI*, Morgan Kaufmann, Los Altos., 1987

[5] M. Aurnague: " Contribution a l'etude de la semantique formelle de l'espace et du raisonnement spatial: la localisation interne en francais, semantique et structures inferentielles", PhD thesis, l'Universite Paul Sabitier de Toulouse, 1991.

[6] B. L. Clarke: "A Calculus of Individuals Based on Connection," *Notre Dame Journal of Formal LOGIC*, vol. 2, No. 3., 1981

[7] B. L. Clarke: "Individuals and Points," *Notre Dame Journal of Formal Logic* Vol. 26, No. 1, 1985

[8] A. G. Cohn: " A More Expressive Formulation of Many Sorted Logic," *J. Autom. Reasoning.*, Vol. 3(2), pp. 113–200, 1987

[9] A. G. Cohn: "Completing Sort Hierarchies," *Computers and Mathematics with Applications*, 1992

[10] Z. Cui, A. G. Cohn and D. A. Randell: " Qualitative Simulation Based On A Logical Formalism Of Space And Time," in proc. of AAAI92, 1992

[11] C. Freksa: "Qualitative Spatial Reasoning", Workshop RAUM, University of Koblenz, 1990.

[12] A. Galton: "A critical examination of Allen's theory of action and time," AI Journal, vol 42 pp 159-188 1990

[13] C. L. Hamblin: " Starting and Stopping," *The Monist.*, 410-425, 1967

[14] C. L. Hamblin: " Instants and Intervals," *Studium Generale.*, Vol. 24, pp. 127–134, 1971

[15] D. Hernandez: "Using Comparative Relations to represent Spatial Knowledge", Workshop RAUM, University of Koblenz, 1990.

[16] T. de Laguna: "Point, Line and Surface as sets of Solids" *The Journal of Philosophy.*, Vol 19., pp. 449–461, 1922.

[17] D. Randell, A. G. Cohn and Z. Cui: "Naive Topology: modeling the force pump," *in Recent Advances in Qualitative Reasoning*, ed B Faltings and P Struss, MIT Press, in press, 1992.

[18] D. Randell, Z. Cui and A. G. Cohn: "An Interval Logic for Space based on "Connection"," *in proc. of ECAI92, 1992*

[19] D. A. Randell: "Analysing the Familiar: Reasoning about space and time in the everyday world," *PhD Thesis*, University of Warwick, UK 1991

[20] D. A. Randell, A. G. Cohn and Z. Cui: "Computing Transitivity Tables: a Challenge for Automated Theorem Provers," in proc. of CADE11, 1992

[21] D. Randell and A. G. Cohn: "Modelling Topological and Metrical Properties in Physical Processes," in *Principles of Knowledge Representation and Reasoning*, ed. R. J. Brachman, H. Levesque and R. Reiter, Morgan Kaufmann, Los Altos. 1989

[22] D. A. Randell and A. G. Cohn: "Exploiting Lattice in a Theory of Space and Time," *Computers and Mathematics with Applications*, 1992

[23] A. Tarski: "Foundations of the geometry of solids," in Logic, Semantics, Metamathematics, trans. J. H. Woodger, Oxford University Press, Oxford, 1956

[24] L. Vieu: "Semantique des relations spatials et inference spatio-temporelles: Une contribution a l'etude des structures formelles de l'espace en Language Naturel.", PhD thesis, l'Universite Paul Sabitier de Toulouse, 1991.

[25] A. N. Whitehead: "Process and Reality: Corrected Edition," eds. D.R. Griffin and D.W. Sherburne, The Free Press, Macmillan Pub. Co., New York, 1978

Axiomatizing Qualitative Process Theory

Ernest Davis
Courant Institute
New York, New York

Abstract

We show that the type of reasoning performed by Forbus' (1985) Qualitative Process (QP) program can be justified in a first-order theory that models time and other measure spaces as real-valued quantities. We consider the QP analysis of a can of water with a safety valve being heated over a flame. We exhibit a first-order theory for the microworld involved in this example, and we prove the correctness of the first two transitions in the envisionment graph. We discuss the possibility of deriving the closure conditions in the theory via non-monotonic inference.

One way to increase confidence in a reasoning program is to show that its reasoning corresponds to valid inferences in a logical theory. Such a correspondence has been shown for many physical reasoners. The calculations of QSIM (Kuipers 1986) correspond to theorems in real analysis; (Duchier 1991) exhibits first-order proofs of these. Likewise the reasoning in NEWTON (de Kleer 1977) and ENVISION (de Kleer and Brown 1985) can be shown to be valid for a simple physical theory, easily formalized in first-order logic, in which time and other physical parameters are viewed as real-valued quantities (Rayner 1991), (Davis 1990).

However, no adequate logical analysis has hitherto been given for the Qualitative Process (QP) program (Forbus 1985). In QP, processes start and end, and the physical system changes its topology over time. Hence, formulating the needed closed-world assumptions on processes and influences has seemed daunting. This paper closes the gap by characterizing inference in QP in a monotonic theory based on real analysis. There are two key points:

- For each parameter, the theory must give an exhaustive enumeration of the types of processes and parameters that can influence it. These axioms resemble to the kind of frame axioms advocated by Schubert (1991), which give necessary conditions for a fluent to change its value. They are also analogous to the circumscription over causes of change discussed in (Lifschitz 1987).
- Since QP uses only qualitative information as to the direction of change and influence, it is possible to combine influences using only existential criteria. A parameter may change in some direction if some influence pushes it in that direction. It must change in a direction if some influence pushes it in that direction, and no influence pushes it in the opposite direction.

Though QP theory centers on continuous parameters, these theories can be extended to include discrete change as well, as we shall show below.

Let us clarify the relation of the axiomatization here to QP. There is essentially no knowledge of physics built into QP. Rather, the QP representation gives a language in which (certain) physical theories can be expressed and associated physical situations can be described; and the QP algorithm uses the information to predict physical behavior over time. The user of QP must input both the specific scenario and also the physical theory to be used. Thus, the axioms that are common across all uses of QP are limited: they include the axioms of real analysis, some basic axioms of temporal reasoning, and a few general axioms constraining the possible behavior of a physical parameter, and relating it to the influences on it. The other physical knowledge needed is all part of the user specification.

In this paper, we show that one particular physical QP theory and one associated scenario can be expressed axiomatically, and that the inferences derived by the QP program can be justified as sound inferences from the axiomatization. It should be clear by analogy how other QP theories and scenarios could likewise be translated into axiomatic form. We do not, however, attempt to give a complete, formal characterization of how such a translation could be done in general. Such a characterization would be needed for a formal verification of the correctness of the QP program. However, it is not needed for our purpose in this paper, which is

to show how theories and inferences like those in QP can be expressed and justified in formal logic.

This paper is organized as follows. Section 1 provides a high-level view of the axiomatics. Section 2 deals with some fine points in defining certain properties of real-valued parameters. Sections 3 and 4 give a detailed analysis of a simple physical system combining continuous and discrete components: a boiling can of water with a safety valve. Section 3 presents a general language for QP theory and an axiomatization of the particular microworld used for this example. Section 4 specifies the particular scenario and shows that the predictions of QP theory can be justified in the logic. Section 5 discusses the application of non-monotonic logic to this theory. Section 6 discusses some features of the theory, and presents the conclusions.

1 Structure of the Theory

The ontology of QP follows familiar lines. The time line and other measure space, such as temperature and mass, are taken to be isomorphic to the real line. The time line does not branch. (Branching in envisionments is expressed by disjunctive uncertainty in prediction, rather than to actual branching in time.) The logic uses two kinds of temporal entities: ***situations***, which are instants of time, and ***time intervals***, which may be closed or open, bounded or unbounded.

A ***fluent*** associates values in some range with instants of time. A fluent with range {TRUE, FALSE} is called a ***Boolean fluent*** or ***state***. A fluent from time to a measure space is called a ***parameter***. If A is a state and S is a situation, then the predicate "holds(S, A)" means that A is TRUE in S. If F is a fluent other than a state, then the function "value_in(S, F)" gives the value of F in situation S. Alternatively, as a notational convenience, if term $\tau(\alpha_1 \ldots \alpha_k)$ denotes a fluent, we may add the situation as an additional argument, in the form $\tau(\alpha_1 \ldots \alpha_k, S)$. This will mean the same as either "holds$(S, \tau(\alpha_1 \ldots \alpha_k))$", if τ is a state, or as "value_in$(S, \tau(\alpha_1 \ldots \alpha_k))$", if τ is not a state. For example, we may say that Valve 1 is open in situation s0 either in the form "holds(s0, open(valve1))" or in the form "open(valve1,s0)".

A function or a predicate defined on a particular space may be extended in the natural way to take arguments that are fluents with range in that space. For example, if "square(X)" is a function mapping the reals to the reals, and F is a real-valued fluent, then "square(F)" is the fluent that, at any given instant gives the square of the value of F at that instant. If ">" is a predicate with two real valued arguments and $F1$ and $F2$ are real-valued fluents, then "$F1 > F2$" is the state that holds whenever the value of $F1$ is greater than the value of $F2$.

value_in$(S$,square$(F))$ = square(value_in(S, F)).
holds$(S$,$F1 > F2) \Leftrightarrow$
value_in$(S, F1) >$ value_in$(S, F2)$.

Equality and inequality are exceptions to this. "$F1 = F2$" and "$F1 \neq F2$" are sentences, stating that $F1$ is the same fluent as $F2$, or $F1$ is a different fluent from $F2$, respectively. The state of the current value of $F1$ being equal to the current value of $F2$ is denoted "eql$(F1, F2)$"; the state of the two values being different is denoted "neql$(F1, F2)$".

A ***process*** is a particular category of state. For processes, we use the special predicate "active(S, P)" (process P is active in situation S); this is synonymous with "holds(S, P)" Besides processes, there are ***events***, which occur over finite, non-point, intervals. We write "occur(I, E)" to mean that event E occurs over interval I. In this theory, we deal only with state, fluent, process, and event ***types***, rather than tokens.

Finally, there are physical objects. We use this term loosely to include practically any entity of physical interest that does not fall into the other categories. For example, in modelling water flowing through a tank, one object could be a particular "piece" of water that comes in at one time and goes out at another; another object could be "the water in the tank", which has a mass that changes over time.

A axiomatic QP theory contains axioms of the following forms:

1. **Process definitions.** Necessary conditions and sufficient conditions (they need not be the same) for a process of a given type to be active in a given situation.
2. **Direct Influences.** For each parameter that is influenced directly, an exhaustive enumeration of the processes that influence it, with the directions of influence.
3. **Indirect Influences.** For each parameter that is influenced indirectly, an exhaustive enumeration of the parameters that influence it, with the directions of influence.
4. **General axioms of influence.** Two axioms relating the behavior of a parameter to the influences on it:
   A. A parameter F can only change in direction G (up or down) if there is some influence on F pushing it in direction G.
   B. A parameter F must change in direction G if there are influences on F in direction G, and there are no influences on F in direction $-G$.
5. **Well-behavedness conditions.** Parameters must observe certain constraints as functions of time. Parameters and their first derivatives must be piecewise continuous; they cannot asymptotically approach a value without attaining it;

and order relations between parameters may not change infinitely often in any finite interval (Davis 1992).

6. **Unique names axioms.** Axioms specifying that objects, processes, and parameters with different names are unequal.
7. **Real analysis.** The theory of the real line.
8. **Discrete fluents:** In theories with discrete fluents, necessary conditions and sufficient conditions for each discrete fluent to change its value.
9. **Events:** In theories with events, necessary conditions and sufficient conditions for each event to occur.
10. **Discontinuities:** For each parameter, necessary conditions and sufficient conditions for the parameter to be discontinuous in a given situation. These are analogous to the axioms governing discontinuous velocity in (Rayner 1991).

The example in this paper contains the discrete fluent of the valve being open or closed, but it contains no events or discontinuous parameters. Thus, it contains axioms of category (8), but none of category (9), and in category (10) only the axiom that all parameters are always continuous.

2 The meaning of "direction of change"

There is a technical problem in defining what it means for parameter to be "increasing," "decreasing," or "constant" at an instant of time. Most of the literature on qualitative reasoning uses the sign of the derivative of the parameter at the instant, which works as long as everything can be assumed to be everywhere differentiable. However, this assumption does not seem reasonable within all domains we would like to address in QP. Consider, for example, cutting a string supporting a weight at time $t = 0$. The acceleration changes instantaneously (up to the precision of the model) from 0 to $-g$, so the velocity is not differentiable at $t = 0$.

How we should characterize its behavior at $t = 0$ depends on how we characterize the state of the string. If the string is whole for $t < 0$ and broken for $t \geq 0$, then we should say that the downward velocity is increasing at $t = 0$; if the string is whole for $t \leq 0$ and broken for $t > 0$, then we should say that the velocity is constant at $t = 0$. How we want to characterize the string may in turn depend on considerations external to the QP analysis, such as the geometric theory.

One might be tempted, from this example, to refuse to deal with characterizing behavior at an instant, and demand that characterizations refer to open intervals. But that will hardly do. Very often, parameters are in a constant state only for an instant, such as a ball thrown in the air at the top of its path. Avoiding this would necessarily create a lot of clumsiness.

The solution we propose is as follows: Assume that every parameter is differentiable at every instant both from the right and from the left. We define the "true derivative" to be, disjunctively, either the derivative from the right or the derivative from the left. The disjunction allows the logic to "pick" whichever value will fit in better with the rest of the theory. "Increasing," "decreasing," and "constant" are then defined in terms of the sign of the "true" derivative. Thus, the downward velocity of the weight may be either increasing or constant at $t = 0$, whichever fits better with the rest of the world state. It cannot be decreasing, though.

Note that this means that there can be two parameters $F1$ and $F2$ that are always equal, but $F1$ is increasing at a time that $F2$ is decreasing. Thus, "increasing" and "decreasing" are properties of physical parameters, not of the associated functions of time.

3 The axiomatic theory

This section gives an axiomatic theory for the following example: A can of water is heated over a flame. The can has a safety valve with two states, open and closed. The valve opens when the pressure in the can exceeds a certain fixed pressure; it closes when the pressure drops below another (lower) fixed pressure. The processes we will model are the heat flow from the flame to the can, the heat flow from the can to its contents, the boiling of the water, and the flow of steam from the can through the safety valve to the outside air. We treat the flame as a heat reservoir, capable of supplying arbitrary heat-flow without being affected, and the outer air as a gas reservoir, capable of absorbing arbitrary gas-flow. We ignore the heat flow to the outside air. We make the idealization that water changes from liquid to gas only during a boiling process.

We use a sorted first-order logic with equality. The sorts of variables is indicated by the first letter of the variable name. We use the following sorts: situations (S), real-numbers (X, Y), signs (G), parameters (F), processes (P), states(A), objects(O). The signs are "pos", "neg" and "zero". For a microworld with events, it would also be necessary to include events and intervals.

The theory below contains only the physics needed for this particular example, and thus does not satisfy the "no function in structure" principle. Obvious extensions within the same general microworld, such as the processes of melting, freezing, condensing, or liquid flow, have not been included. However, it can be seen that these could be added with minor modifications to the analysis of this example.

3.1 Formal Language

The following non-logical primitives are used. Predicate symbols are labelled "[P]"; function symbols are labelled "[F]".

Arithmetic: We use the basic arithmetic functions and the order relations with their usual meaning. We use "pos", "neg", and "0" for the three signs; the arithmetic functions are extended to these in the usual way.

General properties of parameters and states:
holds(S, A). State A holds at time S. [P]
value_in(S, F). Value of parameter F at time S. [F]
one_side_deriv(F, S, X, G). F is differentiable from the side indicated by sign G at time S, and the derivative from that side is X. [P]
direction(F). The fluent of the sign of the direction in which F is changing. [F] constant.) [F]
good_param(F) [P] F is a well-behaved parameter.

Influence:
d_influence(P, F). The fluent of the sign of the direct influence of process P on parameter F in each situation. 0 if no influence. [F]
i_influence($F1, F$). The fluent of the sign of the indirect influence of parameter $F1$ on parameter F in each situation. 0 if no influence. [F]
influence(Q, F, S). The fluent of the sign of the net influence of Q on parameter F in situation S. Q may be either a process or another parameter. [F]
direct_influence(F). F is the sort of parameter that is subject to direct rather than indirect influences. [P].

Invariants:
boiling_point(O). Boiling temperature of object O. [F]
heat_reservoir(O). O is a heat reservoir. [P]
gas_reservoir(O). O is a gas reservoir. [P]
vconn($OV, O1, O2$). OV is a valve connecting $O1$ with $O2$ [P].
tconn($O1, O2$). $O1$ is thermally connected to $O2$. [P]
open_diff(OV). Pressure difference sufficient to open valve OV. [F]
close_diff(OV). Pressure difference small enough that valve OV will close itself [F].

Parameters: temp(O), heat(O), press(O), lmass(O), gmass(O). These are functions mapping an object to parameters: respectively, its temperature; its heat; the pressure of its gaseous part; the mass of its liquid part; and the mass of its gaseous part. [F]

Object States:
open(O). State of valve O being open. [F]
conduit($OC, O1, O2$). State of OC serving as a conduit connecting $O1$ with $O2$. [F]

Processes:
heat_flow($O1, O2$). Process of a heat flow from $O1$ to $O2$. [F]
boiling(O). Process of object O boiling. [F]
gas_flow($O1, O2, OC$). Process of a flow of gas from $O1$ to $O2$ through conduit OC. [F]

Envisionments: These are primitives that are useful in describing envisionments. They are not used either in the axioms describing the microworld or in the axioms describing the scenario. They are defined in axioms E.1-E.4. Envisionments are described in terms of "modes", which are states. We use variables with initial letter M for modes.

throughout($S1, S2, A$). State A holds over the *open* interval ($S1, S2$). [P]
dense($S1, S2, A$). State A holds over a dense subset of the interval ($S1, S2$). [P]
borders(MA, MB, S). Mode MA borders mode MB in situation S. [P]
transition($M0, T, M1, M2 \ldots Mk$). Mode $M0$ may transition to one of $M1 \ldots Mk$. If the Boolean argument T is "terminal," then $M0$ may be a terminal state; otherwise it cannot be. [P]

3.2 Microworld Theory

We now enumerate the axioms for our microworld, organized according to the outline in section 1.

1. Process Definitions: We include here a number of atemporal axioms and state coherence axioms (axioms constraining the states that can hold in a single situation) constraining relations and states strongly associated with activation conditions.

1.1 [tconn(OS, OD) $\wedge$
temp(OS, S) > temp(OD, S)] $\Rightarrow$
active(S,heat_flow(OS, OD)).
(Sufficient condition for heat flow: If source OS is thermally connected to destination OD and OS is hotter than OD, then heat will flow from OS to OD.)

1.2 active(S, heat_flow(OS, OD)) $\Rightarrow$
[$OS \neq OD \wedge$ tconn(OS, OD) $\wedge$
temp(OS, S) $\geq$ temp(OD, S) $\wedge$
$\neg$active(S,heat_flow(OD, OS))]
(Necessary conditions for heat flow: For heat to flow directly from OS to OD, they must be thermally connected; OS must be at least as hot as OD; and there must not be heat flow in the other direction.)

1.3 tconn($O1, O2$) $\Leftrightarrow$ tconn($O2, O1$).
(Thermal connections are symmetric.)

1.4 active(S,boiling(OB)) $\Leftrightarrow$
[lmass(OB, S) > 0 $\wedge$
temp(OB, S) = boiling_point(OB) $\wedge$
direction(heat(OB),S) = pos.]
(Necessary and sufficient conditions for boiling: An object OB will boil iff it is partially liquid and is at its boiling point and its heat is increasing.)

1.5 $\text{lmass}(OB,S) > 0 \Rightarrow$
$\text{temp}(OB,S) \leq \text{boiling\_point}(OB)$.
(Constraint: An object can be partially liquid only if its temperature is below the boiling point.)

1.6 $\text{active}(S,\text{gas\_flow}(O1,O2,OC)) \Leftrightarrow$
$[\text{conduit}(OC,O1,O2,S) \wedge \text{gmass}(O1,S) > 0 \wedge \text{press}(O1,S) > \text{press}(O2,S)]$
(Necessary and sufficient condition for gas-flow: Gas flows from $O1$ to $O2$ through OC if and only if OC is a conduit between $O1$ and $O2$, and $O1$ is partially gaseous, and the pressure in $O1$ is greater than that in $O2$.)

1.7 $\text{conduit}(OC,O1,O2,S) \Leftrightarrow$
$\text{conduit}(OC,O2,O1,S)$.
(The conduit relation is symmetric in the two ends.)

1.8 $\text{lmass}(O,S) \geq 0 \wedge \text{gmass}(O,S) \geq 0$.
(Masses are non-negative.)

1.9 $\text{gmass}(O) = 0 \Rightarrow \text{press}(O) = 0$.
(If there is no gas, there is no pressure.)

2. Direct Influences

2.1 $[\text{direct\_influence}(F) \Rightarrow \text{i\_influence}(F,S)=0] \wedge$
$[\neg\text{direct\_influence}(F) \Rightarrow \text{d\_influence}(F,S)=0] \wedge$
$[\text{direct\_influence}(F) \Leftrightarrow$
$\exists_O\ F=\text{heat}(O) \vee F=\text{lmass}(O) \wedge F=\text{gmass}(O)]$.
(Division of parameters into those that are directly influenced and those that are indirectly influenced, and an enumeration of the directly influenced.)

2.2 $\text{d\_influence}(P,\text{heat}(O),S) = \text{pos} \Leftrightarrow$
$\neg\text{heat\_reservoir}(O) \wedge \exists_{O1}\ P=\text{heat\_flow}(O1,O)$
(Heat in objects that are not reservoirs is increased by incoming heat flow, and nothing else.)

2.3 $\text{d\_influence}(P,\text{heat}(O),S) = \text{neg} \Leftrightarrow$
$\neg\text{heat\_reservoir}(O) \wedge \exists_{O1}\ P=\text{heat\_flow}(O,O1)$
(Heat in objects that are not reservoirs is decreased by outgoing heat flow, and nothing else.)

2.4 $\neg\text{d\_influence}(P,\text{lmass}(O),S) = \text{pos}$.
(There are no processes, within this theory, that tend to increase liquid mass.)

2.5 $\text{d\_influence}(P,\text{lmass}(O),S) = \text{neg} \Leftrightarrow$
$P=\text{boiling}(O)$
(Liquid mass is decreased by boiling, and nothing else.)

2.6 $\text{d\_influence}(P,\text{gmass}(O),S) = \text{pos} \Leftrightarrow$
$[\neg\text{gas\_reservoir}(O) \wedge$
$[P=\text{boiling}(O) \vee$
$\exists_{O2,OC}\ P=\text{gas\_flow}(O2,O,OC)]]$
(Gas mass is increased by boiling and by incoming flow.)

2.7 $\text{d\_influence}(P,\text{gmass}(O),S) = \text{neg} \Leftrightarrow$
$\neg\text{gas\_reservoir}(O) \wedge$
$\exists_{O2,OC}\ P=\text{gas\_flow}(O,O2,OC)$
(Gas mass is decreased by outgoing flow.)

3. Indirect influences

3.1 $\text{i\_influence}(F,\text{temp}(O),S) = \text{pos} \Leftrightarrow$
$F=\text{heat}(O) \wedge \neg\text{active}(S,\text{boiling}(O))$
(Heat is a positive influence on temperature, as long as the object is not boiling.)

3.2 $\neg\text{i\_influence}(F,\text{temp}(O),S) = \text{neg}$.
(There are no negative indirect influences on temperature.)

3.3 $\text{i\_influence}(F,\text{press}(O),S) = \text{pos} \Leftrightarrow$
$[F=\text{gmass}(O) \vee$
$[\text{gmass}(O,S) > 0.0 \wedge F=\text{temp}(O)]]$
(Heat and gaseous mass are positive influences on pressure.)

3.4 $\neg\text{i\_influence}(F,\text{press}(O),S) = \text{neg}$.
(There are no negative indirect influences on pressure.)

4. General axioms of influence.

4.1 $\text{influence}(Q,F,S)=G \Leftrightarrow$
$[[\text{direct\_influence}(F) \wedge \text{active}(S,Q) \wedge$
$\text{d\_influence}(Q,F,S)=G] \vee$
$[\neg\text{direct\_influence}(F) \wedge$
$G=\text{i\_influence}(Q,F) \cdot \text{direction}(Q,S)]]$
(Definition: Q influences parameter F in direction G in situation S if Q is a process active in S that directly influences F in direction G, or if Q is a parameter whose change in S indirectly influences F in direction G.)

4.2. $G=\text{direction}(F,S) \neq 0 \Rightarrow$
$\exists_Q\ \text{influence}(Q,F,S) = G$.
(A parameter F can only change in direction $G \neq 0$ (pos or neg) if there is some influence on F pushing it in direction G.)

4.3 $[\exists_Q\ \text{influence}(Q,F,S)=G \wedge$
$\neg\exists_Q\ \text{influence}(Q,F,S)=-G] \Rightarrow$
$G=\text{direction}(F,S)$.
(A parameter F must change in direction G if there are influences on F in direction G, and there are no influences on F in direction $-G$.)

5. Well-behavedness conditions: Parameters in any QP domain are assumed to be well-behaved: piecewise continuous; differentiable everywhere from the right and from the left; attaining any value approached asymptotically; and not changing order relations infinitely often in finite period. In the domain of our example, we make the further assumption that parameters are everywhere continuous. These properties are abbreviated in the predicate "good_param(F)". The longer version of this paper [Davis, 91] contains detailed mathematical axiomatizations of these properties.

5.1 $\forall_F\ \text{good\_param}(F)$.
(All of the parameters are well behaved and continuous.)

5.2 good_param(F) ⇒
$\exists_{G,X}$ one_side_deriv(F, S, X, G) ∧
sign(X)=direction(F, S).
(Partially determined definition of direction: F is changing in direction G if G is the sign of either the derivative from the left or from the right. See section 2.)

6. Unique names: Processes and parameters of different names are different entities. We omit the detailed enumeration, particularly since these axioms are not actually needed for our proof.

7. Real analysis: The usual axioms for real analysis. These are not enumerated here.

8. Discrete changes (Valves):

8.1 [vconn($OV, O1, O2$) ∧
press($O1, S$) − press($O2, S$) ≥ open_diff(OV)] ⇒
open(OV, S).
(A valve OV must be open if the pressure difference exceeds the "open pressure.")

8.2 [vconn($OV, O1, O2$) ∧
press($O1, S$) − press($O2, S$) ≤ close_diff(OV)] ⇒
¬open(OV, S).
(A valve OV must be closed if the pressure difference is less than the "close pressure.")

8.3 [$S1 < S2$ ∧ vconn($OV, O1, O2$) ∧ ¬open($OV, S1$)
∧ open($OV, S2$)] ⇒
$\exists_S$ $S1 < S \leq S2$ ∧
press($O1, S$) − press($O2, S$) ≥ open_diff(OV).
(Frame axiom: The valve opens only if the pressure attains the open pressure.)

8.4 [$S1 < S2$ ∧ vconn($OV, O1, O2$) ∧ open($OV, S1$) ∧
¬open($OV, S2$)] ⇒
$\exists_S$ $S1 < S \leq S2$ ∧
press($O1, S$) − press($O2, S$) ≤ close_diff(OV).
(Frame axiom: The valve closes only if the pressure difference falls under the close pressure.)

8.5 0 < close_diff(OV) < open_diff(OV).
(The close pressure is less than the open pressure.)

8.6 vconn($OV, O1, O2$) ⇒.
[conduit($OV, O1, O2, S$) ⇔ open(OV, S)].
(A valve is a conduit for gas flow just if it is open.)

Definitions of Envisionment Primitives: These are definitions of the temporal primitives used in the descriptions of envisionment graphs.

E.1 throughout($S1, S2, A$) ⇔
[$S1 < S2$ ∧ $\forall_S$ $S1 < S < S2$ ⇒ holds(S, A)].
(State A holds throughout the open interval $(S1, S2)$.)

E.2 dense($S1, S2, A$) ⇔
[$\forall_{SA,SB} S1 < SA < SB < S2$ ⇒
$\exists_{SZ}$ $SA < SZ < SB$ ∧ holds(SZ, A)].
(State A holds on a dense subset of $(S1, S2)$.)

E.3 borders(MA, MB, S) ⇔
[[holds(S, MA) ∧ $\exists_{S1>S}$ throughout($S, S1, MB$)]
∨
[holds(S, MB) ∧ $\exists_{S1<S}$ throughout($S1, S, MA$)]]
(In state S, the system goes from mode MA to mode MB.)

E.4 transition($M0, T, M1, M2 \ldots Mk$) ⇔
[$\forall_S$ holds($S, M0$) ⇒
[[T=terminal ∧ $\forall_{SA>S}$ holds($SA, M0$)] ∨
$\exists_{S1}$ [$S1 = S$ ∨ throughout($S, S1, M0$)] ∧
[borders($M0, M1, S1$) ∨ ... ∨
borders($M0, Mk, S1$)]].
(If the system is in mode $M0$ then it may change to mode $M1$ or to mode $M2 \ldots$ or to mode Mk or, if T is "terminal" it may remain in $M0$ forever.)

4 Scenario Description and Envisionment

In this section, we first give a formal account of our sample scenario. Second, we define the first two modes of the system. Third, we prove that mode 1 must be followed by mode 2. (In the more extended paper [Davis, 91] we prove the first two mode transitions of the system.

4.1 Scenario Description

SC.1 in_scenario(O) ⇔
O=oflame ∨ O=ocan ∨ O=owater ∨ O=ovalve ∨
O=outside_air.
(Enumeration of the objects in the scenario. Note: owater is the collective H_2O in the can, both liquid and steam. This decreases as steam is released through the valve.)

SC.2 in_scenario(O) ⇒
[heat_reservoir(O) ⇔ O=oflame].
(The flame is the only heat reservoir.)

SC.3 in_scenario(O) ⇒
[gas_reservoir(O) ⇔ O=outside_air].
(The outside air is the only gas reservoir.)

SC.4 tconn(O,ocan) ⇔ O=oflame ∨ O=owater.
(The flame and the water are the only things thermally connected to the can.)

SC.5 tconn(O,owater) ⇔ O=ocan.
(The can is the only thing thermally connected to the water. We ignore any heat flows involving the valve or the outside air.)

SC.6 vconn(ovalve,owater,outside_air).
(The valve is a valve connecting the water in the can to the outside air.)

SC.7 conduit(OC,owater,OD,S) ⇒
OC=ovalve ∧ OD=outside_air.
(The valve is the only conduit from the water in the can to the outside air. The statement that

the valve is a conduit when open is in axiom 8.6 above.)

SC.8 distinct(oflame, ocan, owater, ovalve, outside_air).
(Unique names.)

SC.9 boiling_point(ocan) > temp(oflame,$S1$) = temp(oflame,$S2$) > boiling_point(owater).
(The temperature of the flame is constant, greater than the boiling point of water, and less than the boiling point of the can.)

SC.10 press(outside_air,$S1$) = press(outside_air,$S2$)
(The pressure of the outside air is constant.)

SC.11 open_pressure =
press(outside_air,S) + open_diff(ovalve).
close_pressure =
press(outside_air,S) + close_diff(ovalve).
(Landmarks on the pressure of the steam in the can to open or close the valve.)

4.2 Mode Definitions

MD.1 holds(S,mode1) ⇔
temp(owater,S) ≤ temp(ocan,S) ≤
temp(oflame,S) ∧
temp(owater,S) < boiling_point(owater) ∧
lmass(owater,S) > 0.0 ∧
gmass(owater,S)=0.0 ∧
press(owater,S) < open_pressure ∧
¬open(ovalve).
(The water is liquid and not boiling, the valve is closed.)

MD.2 holds(S,mode2) ⇔
temp(owater,S) ≤ temp(ocan,S) ≤
temp(oflame,S) ∧
temp(owater,S) = boiling_point(owater) ∧
lmass(owater,S) > 0.0 ∧
press(owater,S) < open_press ∧
¬open(ovalve).
(The water is boiling, the valve is closed.)

4.3 Proof of the first transition

The presence of the two coupled heat flows, from the flame to the can, and from the can to the water, gives rise to complexities in the predictions and the proof. It is perfectly consistent with the above theory that the can should either attain the temperature of the flame, or that it should attain the temperature of the water. (These are achievable states even if axioms 1.1 and 1.2 are changed to read that no heat flow can occur unless there is a temperature differential.) In fact, the temperature of the can can do anything it wants to as long as it stays between the temperature of the flame and the temperature of the water. If the can gets as hot as the flame, then the heat-flow from the flame to the can may cease. It can only cease for an instant, though, because the heat flow from the can to the water will bring down the temperature of the can immediately. Similarly, if the can gets as cool as the water, then the heat flow from the can to the water will cease, and the temperature of the water will stop rising; but, again, this can only happen for an instant. (This problem was called "stutter" in [Forbus, 85].) Therefore, some of our results are stated, not in the form "Such and such a condition must hold throughout an interval," but in the form, "The condition must hold over a dense subset of the interval." It is possible to prove mathematically that these conditions must, in fact, hold almost everywhere on the interval. However, since this stronger conclusion does not give us any additional leverage, we have not included it in the proof below.

Lemmas of purely mathematical content are merely stated and not proven.

Lemma 1:
temp(owater,S) < temp(ocan,S) ⇒
active(S,heat_flow(ocan,owater)).
(If the water is cooler than the can, there must be a heat flow from the can to the water.)

Proof: 1.1, SC.5. □

Lemma 2:
temp(ocan,S) < temp(oflame,S) ⇒
active(S,heat_flow(oflame,ocan)).
(If the can is cooler than the flame, there must be heat flow from the flame to the can)

Proof: 1.1, SC.4. □

Lemma 3:
[good_param($F1$) ∧ good_param($F2$) ∧
throughout($S1, S2$,eql($F1, F2$))] ⇒
dense($S1, S2$,eql(direction($F1$),direction($F2$)))
(Mathematical. If parameters $F1$ and $F2$ are equal throughout the interval $(S1, S2)$ then their directions have to be equal on a dense subset.)

Lemma 4:
[good_param(F) ∧
throughout($S1, S2$,eql(direction(F),0))] ⇔
$\exists_X$ throughout($S1, S2$,eql(F, X)).
(Mathematical: A parameter is constant over an open interval just if its direction is always 0.)

Lemma 5:
throughout($S1, S2$,eql(temp(oflame),temp(ocan))) ∧
throughout($S1, S2$,temp(ocan) > temp(owater)) ⇒
throughout($S1, S2$,heat_flow(oflame,ocan)).
(If the can and the flame are the same temperature and hotter than the water throughout an open interval, then there must be heat flow from the flame to the can throughout the interval. Note: This does not apply to a closed interval.)

Proof: By Lemma 1, there is a heat-flow from the can to the water. By 2.3, SC.1, SC.2, this is a negative influence on heat(ocan).

By SC.9. temperature(oflame) is constant, so, by assumption, temperature(ocan) is likewise constant. By Lemma 4, the direction of temperature(ocan) is 0. By SC.9 and 1.4, the can is not boiling. By 3.1, 3.2, heat(ocan) is an influence and the only influence, on temperature(ocan). By 4.1, 4.3, the direction of heat(ocan) is 0. Since we know that there is a negative influence on heat(ocan), by 4.3, there must be a positive influence on heat(ocan). By 2.2, this must be a heat flow into ocan. By 1.2 and SC.4, the only possible heat flow into ocan is from oflame.□

Lemma 6:
direction(temp(O),S)=pos ⇒
$\exists_{O1}$ active(S,heat_flow($O1$,O)).
(The temperature of O can increase only if there is a heat flow into it.)

Proof: From 3.1, 3.2, 4.1, 4.2, the temperature of O can increase only if the heat of O increases. From 2.2, 4.1, 4.2, heat(O) can increase only if there is a heat flow into O.□

Lemma 7:
direction(temp(O),S)=neg ⇒
$\exists_{O1}$ active(S,heat_flow(O,$O1$)).
(The temperature of O can decrease only if there is a heat flow out of it.)

Proof: From 3.1, 3.2, 4.1, 4.2, the temperature of O can decrease only if the heat of O decreases. From 2.3, 4.1, 4.2, heat(O) can decrease only if there is a heat flow out of O.

Lemma 8:
throughout($S1$, $S2$,temp(oflame) > temp(ocan)) ∧
throughout($S1$, $S2$,eql(temp(ocan), temp(owater))) ⇒
$\exists_S$ $S1 < S < S2$ ∧ active(S,heat_flow(ocan,owater)).
(If the can and the water are the same temperature and cooler than the flame throughout an open interval, then there is heat flow from the can to the water at some time during that interval.)

Proof: By Lemma 2, there is a heat flow from oflame to ocan. By 1.2 and SC.4, the only possible heat flow out of ocan is to owater.

We prove by contradiction that at some time between $S1$ and $S2$ there must be a heat flow from ocan to owater. Suppose not. Then, from the above remark, there is no heat flow out of ocan. By 2.3, there is no negative influence on the heat of ocan, and by 2.2 there is a positive influence. By 4.3, the heat of ocan is rising throughout the interval ($S1$, $S2$). By SC.9 and 1.4, the can is not boiling, so by 3.1 and 3.2, the heat of the can is the unique influence on temperature. Therefore, by 4.3, the temperature of the can rises throughout ($S1$, $S2$). By Lemma 3, since the assumptions specify that the temperature of ocan and owater are equal throughout ($S1$, $S2$), it follows that the temperature of owater is rising at a dense subset of ($S1$, $S2$). By lemma 6, there must be a heat flow into owater from somewhere. By 1.2 and SC.5, the only possible source for a heat flow into owater is ocan; but by assumption there is no such heat flow. This completes the contradiction.□

Lemma 9:
[throughout($S1$, $S2$,temp(oflame) ≥ temp(ocan)) ∧
throughout($S1$, $S2$,temp(ocan) ≥ temp(owater)) ∧
throughout($S1$, $S2$,temp(oflame) > temp(owater))] ⇒
$\exists_S$ $S1 < S < S2$ ∧ active(S,heat_flow(oflame,ocan)) ∧
active(S,heat_flow(ocan,owater)).
(If the temperature of the can is (not strictly) between the temperature of the flame and the temperature of the water throughout an open time interval, then at some time in between there must be both heat flow from the flame to the can, and heat flow from the can to the water.)

Proof: There must be some subinterval SA, SB of $S1, S2$ throughout which one of the following holds:

- temp(oflame) > temp(ocan) > temp(owater).
  By lemmas 1 and 2, the two heat flows are active.
- temp(oflame) = temp(ocan) > temp(owater).
  By lemmas 1 and 5, the two heat flows are active.
- temp(oflame) > temp(ocan) = temp(owater).
  By lemmas 2 and 6, the two heat flows are active.□

Lemma 10:
[$\forall_{SA,SB}$ throughout($SA, SB, A1$) ⇒
$\exists_S$ $SA < S < SB$ ∧ holds(S, $A2$)] ⇒
[$\forall_{SA,SB}$ throughout($SA, SB, A1$) ⇒
dense($SA, SB, A2$)].
(Mathematical. If every interval satisfying $A1$ throughout contains a point satisfying $A2$, then every interval satisfying $A1$ contains a dense collection of points satisfying $A2$.)

MODE1.1:
throughout($S1$, $S2$,mode1) ⇒
dense($S1$, $S2$,heat_flow(oflame,ocan)) ∧
dense($S1$, $S2$,heat_flow(ocan,owater)).
(If mode 1 holds throughout an interval, then there is heat flow from the flame to the can and from the can to the water over a dense subset.)

Proof: Immediate from MD.1, Lemmas 9 and 10. □

Lemma 11:
[active(S,heat_flow(ocan,owater)) ∧
¬active(S,boiling(owater))] ⇒
direction(temp(owater,S)) = pos.
(If there is a heat-flow from the can to the water, and the water is not boiling, then the temperature of the water is rising.)

Proof: By 1.2 and SC.5, there cannot be any heat flow out of the water. By 2.2, 2.3, 4.1, and 4.3, heat(owater) must be increasing. By 3.1, 3.2, this is the only influence on the temperature of the water. By 4.1 and 4.3

temp(owater) must be rising.□

Lemma 12:
good_param(F) ∧ $G \neq 0$ ∧
dense($S1, S2$,eql(direction(F),G)) ⇒
throughout($S1, S2$,neql(direction(F),$-G$)).
(Mathematical: If direction(F) has non-zero value G over a dense subset of $(S1, S2)$, then it cannot be $-G$ anywhere on $(S1, S2)$.)

MODE1.2:
throughout($S1, S2$,mode1) ⇒
dense($S1, S2$,eql(direction(temp(owater)),pos)) ∧
throughout($S1, S2$,neql(direction(temp(owater)),neg))
(In mode 1, the temperature of the water is increasing over a dense set, and it is never decreasing.)

Proof: MODE1.1, MD.1, Lemmas 11 and 12. □

Lemma 13:
[good_param($F1$) ∧ good_param($F2$) ∧ $S1 < S2$ ∧
value_in($S1, F1$) ≤ value_in($S1, F2$) ∧
value_in($S2, F1$) > value_in($S2, F2$)] ⇒
$\exists_{SA}$ $S1 < SA < S2$ ∧
value_in($SA, F1$) > value_in($SA, F2$) ∧
[direction($F1, SA$)=pos ∨ direction($F2, SA$)=neg]
(Mathematical: If $F1 \leq F2$ at time $S1$ but $F1$ is greater than $F2$ later, then there is a time later when $F1$ is greater than $F2$ and either $F1$ is increasing or $F2$ is decreasing.)

Lemma 14:
temp(oflame,S) ≥ temp(ocan,S) ≥ temp(owater,S) ⇒
$\forall_{S1>S}$ temp(oflame,$S1$) ≥ temp(ocan,$S1$) ≥ temp(owater,$S1$).
(If the temperature of the can is (not strictly) between the temperature of the flame and the temperature of the water, then these inequalities will hold at all future times.)

Proof: By contradiction. Suppose that this does not hold for some particular S and $S1$. Then in $S1$ either (a) the water is hotter than the can; or (b) the water is not hotter than the can, but the can is hotter than the flame. We consider these two possibilities in turn.

A) By lemma 13, there is some time SA between S and $S1$ during which the water is hotter than the can and the temperature of the water is increasing. But (lemma 6) the temperature of the water can increase only if there is heat flow into the water, which is impossible by 1.2 and SC.5.

A) By lemma 13, there is some time SA between S and $S1$ during which the can is hotter than the flame and the temperature of the can is increasing. But (lemma 6) the temperature of the can can only increase if there is heat flow into the can, which means (1.2 and SC.4) that the water must be hotter than the can, contrary to assumption.

This completes the contradiction.□

Lemma 15:
temp(owater,S) < boiling_point(owater) ⇒
direction(lmass(owater),S) =
direction(gmass(owater),S) = 0.
(If the water is cooler than boiling temperature, then neither liquid mass nor gas mass are changing.)

Proof: From 1.4, the water is not boiling in S. From 2.4, 2.5, 4.1, 4.2, the liquid mass of the water is not changing. From 2.6, 2.7, 4.1, 4.2, the gas mass of the water is not changing either.□

MODE1.3:
holds(S,mode1) ⇒
direction(lmass(owater),S) =
direction(gmass(owater),S) = 0.
(In mode 1, neither liquid mass nor gas mass is changing.)

Proof: Immediate from Lemma 15.□

Lemma 16:
[gmass(owater,$S1$) = 0.0 ∧
[$\forall_S$ $S1 \leq S < S2$ ⇒
temp(owater,S) < boiling_point(owater)]] ⇒
press(owater,$S2$) = press(owater,$S1$).
(If no part of the water is gaseous at $S1$, and the temperature of the water remains below boiling until $S2$, then there is no change in pressure.)

Proof: If the pressure changes between $S1$ and $S2$, then by lemma 4, it must have a non-zero direction at some time in between. From 3.3, 3.4, 4.1, 4.2, the pressure can change only if the gas mass is changing or if the gas mass is greater than 0.0 and the temperature is changing. From Lemma 15, the gas mass is never changing. From lemma 4, this implies that the gas mass remains equal to 0.0 throughout $(S1, S2)$. Thus the result follows.

MODE1.4:
holds(S,mode1) ⇒ direction(press(owater),S) = 0.
(In mode 1 there is no change in pressure.)

Proof: Immediate from lemma 16.□

MODE1.5
holds(S,mode1) ⇒ $\exists_{S>S1}$ ¬holds(S,mode1)
(Mode 1 cannot be a final state.)

Proof: Suppose that mode 1 were a final state. Then, by MODE1.2, the temperature of owater would be forever rising. By definition, a well-behaved parameter that moves forever in one direction approaches infinity, so the temperature of owater would eventually exceed boiling_point(owater). But then the system would no longer be in mode1, which is inconsistent.□

Lemma 17:
good_param(F) ∧ $X1$ < value_in(S, F) < $X2$ ⇒
$\exists_{S1>S}$ throughout($S, S1, X1 < F < X2$).
(Mathematical: If F has value X strictly between $X1$ and $X2$ in situation S, then it continues to lie between

$X1$ and $X2$ for some interval after S.)

Lemma 18:
good_param($F1$) ∧ good_param($F2$) ∧
throughout($S1, S2$, $F1 \leq F2$) ⇒
value_in($S1, F1$) ≤ value_in($S1, F2$) ∧
value_in($S2, F1$) ≤ value_in($S2, F2$).
(Mathematical: If a non-strict inequality holds over an open interval, it holds at both end points.)

Corollary 19:
good_param($F1$) ∧ good_param($F2$) ∧
throughout($S1, S2$, eql($F1, F2$)) ⇒
value_in($S1, F1$) = value_in($S1, F2$) ∧
value_in($S2, F1$) = value_in($S2, F2$).
(If two parameters are equal over an interval, they are equal at the endpoints. Corollary of lemma 18.)

MODE1.6: transition(mode1,nonterminal,mode2).
Mode 1 must transition to mode2.

Proof: By MODE1.5, mode 1 cannot be a final state. Let S be a state in which mode 1 holds, and let $S1$ be the greatest lower bound of all situations greater than S in which mode 1 does not hold. Then mode 1 holds in S and over the open interval $(S, S1)$; but either mode 1 does not hold in $S1$ or there is no interval $(S1, S2)$ such that mode 1 holds throughout $(S1, S2)$.

Let us begin by considering what is the situation in $S1$. If $S1 = S$, then, of course, mode 1 holds in $S1$. Suppose $S1 > S$, so that mode 1 holds throughout the interval $(S, S1)$. By lemma 14, in $S1$ the temperature of the water must still be less than or equal to the temperature of the can, and the temperature of the can must be less than or equal to the temperature of the flame. By lemma 18, the temperature of the water in $S1$ is less than or equal to the boiling point of water. By MODE1.3, MODE1.4, and lemma 4, the liquid mass, the gas mass, and the pressure are all constant over the interval $(S, S1)$, so by lemma 19 they are unchanged in $S1$. Therefore the constraints lmass(owater,$S1$) > 0.0, gmass(owater,$S1$) = 0.0, press(owater,S) < open_pressure must all hold. By 8.3, since the pressure stays less than open_pressure throughout $(S, S1]$, the valve cannot open.

Putting these together, we conclude that in $S1$, either the temperature of the water has reached the boiling point and the system is in mode 2, or it has not and the system is in mode 1. In the first case, there is a transition from mode 1 to mode 2. We will show that the second case is impossible by considering what happens in short intervals following $S1$. By lemma 17, if the temperature of the water is below boiling in $S1$ then there is an interval $(S1, S2)$ during which it remains below boiling. By lemmas 15 and 16, the directions of change of the liquid mass, the gas mass, and the pressure are zero throughout $(S1, S2)$. Hence, by lemma 4, these parameters remain constant. By same argument as above, the valve remains closed throughout $(S1, S2)$. Hence, the system remains in mode 1 throughout $(S1, S2)$, contrary to assumption.□

5 Non-monotonicity

Crawford and Etherington (1992) propose that non-monotonic inferences in predictive reasoning can be divided into two classes:

1. Non-monotonic inferences used in model-building. Example: If a theory does not specify any actions that cause guns to be unloaded, then infer a general rule that there are no actions that cause guns to be unloaded. The non-monotonic inferences in (Sandewall 1989) fall in this category.
2. Non-monotonic inferences used in simulation. Example: If a gun is loaded in situation S, infer that, in this specific situation, the gun will still be loaded after waiting (or after an unspecified period of time.)

To these two categories, it seems reasonable to add a third:

3. Non-monotonic inference in defining the scenario. Examples: Given a description of a blocks-world situation, infer that the only blocks in the situation are those specifically named in the description. Given a description that specifies that "load", "wait", and "shoot" occur during some interval, infer that these are the only events that occur.

An orthogonal categorization divides non-monotonic theories into:

A. Non-monotonic theories that serve, in effect, as abbreviations for monotonic theories. Examples: The causal theories of (Lifschitz 1987) and (Lin and Shoham 1991). The application of the closed-world assumption to databases (Reiter 1978).

B. Non-monotonic theories that express uncertain and defeasible inference. Example: Tweety, being a bird, can fly.

The latter distinction is not a sharp one, and is more a difference of objective and outlook than a technical distinction. On the whole, categories (1) and (3) above tend to be associated with category (A), and category (2) tends to be associated with category (B), though there are many exceptions.

It is easily seen that inferences of categories (1) and (3) and of type (A) can be used to simplify the initial statement of the theory. Clearly many of the closure conditions in the theory and in the scenario description can be omitted and derived via non-monotonic inference of a standard kind. Specifically:

1. When it is possible to state conditions that are both necessary and sufficient for the activity of a process, it will suffice just to state them as sufficient conditions. That they are also necessary can then be derived by circumscribing "active". For example, in axioms 1.4 and 1.6, one could just state the axiom with the left-pointing arrow, and derive the right-pointing arrow by circumscribing "active".

2. In the enumeration of influences, it would be possible just to state axioms of the form "Process P or parameter $F1$ has influence G on parameter $F2$," and then derive that these are the only influences by circumscribing "d_influence" and "i_influence".[1] For example, in the above theory, one would replace axioms 2.2 and 2.3 by the axioms

$$\neg\text{heat\_reservoir}(O) \Rightarrow$$
$$\text{d\_influence}(\text{heat\_flow}(O1, O), \text{heat}(O), \text{pos}).$$
$$\neg\text{heat\_reservoir}(O) \Rightarrow$$
$$\text{d\_influence}(\text{heat\_flow}(O, O1), \text{heat}(O), \text{neg}).$$

3. Every ground instance of the unique-names axioms in the theory and in the scenario description can be derived via the unique-names assumption.

4. It is probably possible to derive the frame axioms 7.3 and 7.4 by choosing a suitable causal language and applying circumscription to the causal axioms 7.1 and 7.2, along the lines of (Lifschitz 1987) and (Lin and Shoham 1991).

5. The exhaustive enumeration of the heat and gas reservoirs in the scenario (SC.2) can be achieved by stating that the flame is a heat reservoir and that the outside air is a gas reservoir, and then circumscribing over those two predicates. Likewise, the exhaustive enumeration of thermal connections (SC.4) can be achieved by stating that the flame is connected to the can, and the can to the water, and then circumscribing over that predicate.

Non-monotonic inference thus allows us to start with a theory that is simpler. It is also more *additive*, in the following sense: If we wish to add a new process to the theory, all that is required is to add axioms describing its activation conditions and its influences and to "re-run" the circumscription. By contrast, adding a new process to the monotonic theory will, in general, require rewriting the closure conditions. Consider, for example, adding "freezing(O)" as a new process and "solid_mass(O)" as a new parameter. In the monotonic theory, axiom 2.6, which states that the only negative influence on liquid_mass(O) is boiling(O), must be weakened to read that the only negative influences are boiling(O) and freezing(O). By contrast, none of the axioms of the non-monotonic theory become false. Likewise, expanding the scenario by adding new thermally connected objects would require rewriting SC.4 and SC.5 in the monotonic theory, but only requires adding the new objects in the non-monotonic theory.

This non-monotonic theory would be considerably stronger than is necessary. For example, circumscribing "i_influence" and "d_influence" would not only generate the needed conclusion that the only influences *on* the parameters of interest are those enumerated; it would also generate the unnecessary conclusion that that the only influences *of* the parameters and processes of interest are those enumerated. Similarly, the application of domain closure to the class of objects would generate the conclusion that the objects enumerated are the only ones that exist; whereas all we need is that they are the only objects involved in the scenario.

Inferences of category (2) and type (B) are more problematic. The modification of the theory so that the closure assumptions are merely defeasible inferences seems attractive in many instances. For example, the assumption that all the relevant influences on a parameter have been enumerated could be made a defeasible inference, that could be withdrawn if the observed behavior of a parameter violated the predicted behavior. If it is observed that the water is not heating up, contrary to prediction, then we must posit that the closure assumption was mistaken and some additional process is active. In terms of circumscription, this would require circumscribing "active" over a theory that included these very observations. However, this kind of inference tends to be prone to anomalies like the Yale Shooting Problem, and a careful analysis would be required to determine whether the theory leads to all and only the reasonable conclusions. (Crawford and Etherington 1992) make some suggestions on this problem.

6 Remarks on the theory

Some particular features of the above theory and inference process are worth noting.

The theory largely achieves the objective of *locality*. It would be possible to posit two separate scenarios running simultaneously side by side, and to reason about them separately. It would even be possible to posit that these same objects were involved in an entire separate collection of parameters and process (e.g. electrical processes). The validity of the proof above would not be affected as long as these additional processes and parameters do not influence our original processes and parameters. (Influence in the opposite direction would be OK.) Nowhere did either the theory or the

[1] This circumscription of d_influence was suggested independently by Yoav Shoham at the IFIP Workshop of Knowledge Representation and Qualitative Reasoning, Islamorada, Fla., Feb. 1992. Shoham also showed that it was possible to derive non-monotonically the condition that the value a parameter depends functionally on the values of its direct influences. However, in our theory we did not need that condition.

scenario description assert that these are the only objects in the world, or that these are the only processes or process types.

As remarked above, the monotonic theory does not have the property of *additivity*, either in expanding the list of processes known to affect a given parameter, or in expanding the list of objects in a scenario. This additivity can be largely achieved, however, if the monotonic theory is derived from an underlying nonmonotonic theory.

The natural form of reasoning in this theory has a somewhat different flavor from the reasoning in QP, even in doing the same task of prediction. QP always starts with a complete qualitative description of some mode, and calculates the next mode. In using the logic, the natural way to proceed is to develop lemmas that start with partial characterizations of a situation or interval, and derive other partial characterizations of that time and future times. For example, lemma 14 posits a constraint on all future situations, given certain characteristics of the starting situation. QP, so to speak, works vertically from one time period to the next; logical inference works most comfortably horizontally, building up constraints among intervals of time. (There is a large overlap among the lemmas used in proving various transitions. For example, once the lemmas above have been established, the proof of the transitions out of mode 2 is only about one third as long as the above proofs of the transitions from mode 1 to mode 2.)

For this reason, certain inferences that require special mechanisms in QP do not require any special treatment in the logic. For example, it is a fact that the cycle of the valve opening and closing while the water boils away cannot persist indefinitely, though it may cycle arbitrarily many times, since the liquid mass drops throughout and must eventually attain zero. This fact cannot even be expressed in a simple envisionment graph. (There are techniques that can be added to QP to handle this (Weld 1986).) However, in the logic, it takes the form of the lemma, "If, throughout an interval interval, the liquid mass is alway positive and always dropping, then the interval must be finite," which is a simple consequence of axiom 5.8.

Acknowledgements

This research was supported by NSF grant #IRI-9001447. This paper was in part inspired by an extended discussion on the QP electronic mailing list in October, 1991. The IFIP Workshop on Knowledge Representation and Qualitative Physics, Islamorada, Fla., Feb. 1992, was extremely helpful in clarifying the role of non-monotonic inference. Thanks to all who participated in these. Thanks also to Denis Duchier for helpful comments.

References

J. Crawford and D. Etherington (1992). "Formalizing Reasoning about Change: A Qualitative Reasoning Approach," *Proc. AAAI-92* (to appear).

E. Davis (1990). *Representations of Commonsense Knowledge,* Morgan Kaufmann, San Mateo, CA.

E. Davis (1991). "Axiomatizing Qualitative Process Theory," NYU Tech. Rep. #590.

E. Davis (1992). "Infinite Loops in Finite Time: Some Observations," these proceedings.

J. de Kleer (1977). "Multiple Representations of Knowledge in a Mechanics Problem Solver," *Proc. IJCAI-77,* pp. 299-304.

J. de Kleer and J.S. Brown (1985). "A Qualitative Physics Based on Confluences," in D. Bobrow (ed.) *Qualitative Reasoning about Physical Systems,* M.I.T. Press, Cambridge, MA.

D. Duchier (1991). "Logicalc: An Environment for Interactive Proof Development," Yale Computer Science Dept., Research Report #862.

K. Forbus (1985). "Qualitative Process Theory," in D. Bobrow (ed.) *Qualitative Reasoning about Physical Systems,* M.I.T. Press, Cambridge, MA.

B. Kuipers (1986). "Qualitative Simulation," *Artificial Intelligence,* vol. 29, pp. 289-338.

F. Lin and Y. Shoham (1991). "Provably Correct Theories of Action," *Proc. AAAI-91,* pp. 349-354.

M. Rayner (1991). "On the applicability of nonmonotonic logic to formal reasoning in continuous time," *Artificial Intelligence,* vol. 49, pp. 345-360.

R. Reiter (1978). "On closed world data bases." In H. Gallaire and J. Minker (eds.) *Logic and Data Bases,* Plenum, New York, pp. 119-140.

E. Sandewall (1989). "Combining logic and differential equations for describing real-world systems," in R. Brachman, H. Levesque, and R. Reiter (eds.) *Proc. First International Conference on Principles of Knowledge Representation and Reasoning,* Morgan Kaufmann, San Mateo, CA, pp. 412-420.

L.K. Schubert (1990). "Monotonic solution of the frame problem in the Situation Calculus: An efficient method for worlds with fully specified actions," in H. Kyburg, R. Loui and G. Carlson (eds.), *Knowledge Representation and Defeasible Reasoning,* Kluwer, pp. 23-67, 1990.

D. Weld (1986). "The Use of Aggregation in Qualitative Simulation," *Artificial Intelligence,* vol. 30, pp. 1-18.

Reasoning with Analogical Representations

Karen L. Myers **Kurt Konolige**
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
myers@ai.sri.com konolige@ai.sri.com

Abstract

Analogical representations have long been of interest to the knowledge representation community. Such representations provide compact encodings of information that can be cumbersome to represent and inefficient to manipulate in sentential languages. In this document, we address the problem of using analogical representations effectively in automated deduction systems. The primary contribution is a formal framework for combining analogical and deductive reasoning. The framework consists of a set of generic operations on analogical structures and accompanying inference methods for integrating analogical and sentential information. The capabilities of the framework are demonstrated for the task of reasoning to extend incomplete maps. The examples presented here have all been solved automatically by an implementation of the integration framework.

1 Introduction

Analogical representations have long been of interest to the knowledge representation community [8, 9, 22, 23]. The attraction of analogical representations lies with their ability to store certain types of information that humans can readily process but are problematic for sentential reasoning systems. Although the power of analogical representations has been acknowledged for many years, little progress has been made in understanding how to exploit the computational advantages that these representations can provide.

Analogical representations encompass both explicit diagrams (as in [6, 7]) and representation structures that are *diagram-like*. Although this latter class is not easily defined, diagram-like representations share with real diagrams the property of certain structural correspondences with the domain being modeled. It is precisely such correspondences that make analogical representations useful. For example, a two-dimensional street map could be represented by graph-theoretic structures in which nodes correspond to intersections and arcs corresponds to road segments. Such a representation is analogical with the world being mapped in two ways. First, paths between nodes in the graph corresponds to road connections in the world being modeled. Second, there is a correspondence between the existence of objects in the world and objects in the representation. For example, all roads are represented in the graph; thus, the closure of the set of roads is implicit. In contrast, expressing such closure information sententially would require an explicit statement that the given roads constitute all roads.

The work described in this paper applies equally well to both diagrams and diagram-like structures. For this reason, we will not distinguish further between the two types. The terms *diagram* and *analogical representation* will be used interchangeably throughout the document.

While analogical representations have received much attention in recent years from psychologists [10, 11, 12], there have been few advances in understanding the computational aspects of analogical reasoning. Until recently, most computationally-oriented work has focused on properties of particular classes of diagrams (*e.g.*, Venn diagrams [21, 20], Euler circles [24], qualitative reasoning [5, 7, 18], geometry [8]), ignoring more general aspects of reasoning diagrammatically. This document addresses the broader question of domain-independent inference techniques for reasoning involving analogical representations. The work encompasses both reasoning *about* and *with* diagrams. The former involves extraction of information from a diagram and amounts to a passive use of diagrams; the latter further supports modifications to diagrams as a result of the reasoning process, thus constituting an active use of diagrams.

Reasoning with and about diagrams should not be accomplished by simply translating the diagram contents

into a sentential language, nor *vice versa*. Analogical structures provide compact representations of information that is cumbersome to express sententially but generally lack the expressive power of sentential languages. Since sentential theories are a more general representational technology, it is tempting to translate analogical structures into first-order sentences *en masse*. But this strategy would compromise the efficiency of the representation system since the specialized inference mechanisms for the analogical structures are replaced by general-purpose deductive methods; this point is borne out by the experimental results of [13, 15]. Here, we adopt a hybrid approach in which separate analogical and sentential subsystems co-exist and inference rules for translating information between the two are defined.

Our hybrid framework is based on a set of generic operations for manipulating analogical structures along with corresponding inference rules that invoke the operations. The operations and rules were chosen for their capacity to increase overall reasoning competency through the appropriate use of analogical information. The framework supports both the incorporation of diagrammatic information into the sentential reasoner and the modification of diagrams to reflect information deduced by sentential reasoning; in other words, both reasoning about and with diagrams.

One particular class of analogical representations to which we apply our work is that of office-building maps. We are currently using an implementation of our framework in the construction of a hybrid map-learning architecture for the SRI mobile robot [16]. For concretenes, we focus on examples from this application; the work, however, applies to all types of analogical structures. Our examples employ schematic map diagrams whose exact representations are left unspecified; the choice of a particular representation is immaterial to the research presented here.

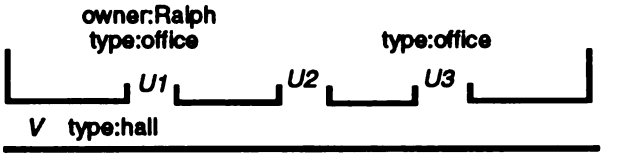
2 The Hybrid Framework

In this section, we describe the analogical and sentential subsystems along with criteria for their integration. Specific integration rules are presented in Sections 3–4.

2.1 Analogical Subsystem

The details of the analogical component will vary for different applications. Our formal framework isolates the integration methods from the specifics of any particular application through the use of an abstract characterization of the information stored in the analogical system. Since we are interested in reasoning with maps, we employ examples from that domain here.

A typical hallway map used by a mobile robot might contain the kind of information displayed in the following diagram:



(1)

The constants V and U_i are symbolic names assigned to the hallway and the three openings on it in the given scene. These objects and the relationships among them are identified by the robot's perceptual interpretation mechanism, which detects relevant geometric properties and segments sensory input into meaningful units (*e.g.*, groups line segments and intersegment spaces into objects such as corridors and significant openings). We use the term *diagram element* for such objects. Prior knowledge about the scene was used to determine the remainder of the information in this diagram, namely that certain U_i are offices and that the leftmost office belongs to Ralph.

For any particular class of applications, there will be a fixed ontology of elements and a fixed set of properties of interest. We consider two classes of properties: symbolic labels for diagram elements and analogical relations among diagram elements. Formally, we can represent the information about labels and relations for diagram elements that is stored in an analogic representation S as a set of first-order models M_S. While a diagram records only those relationships and elements that are known to exist, each of these *diagram models* constitutes a possible completion of the partial information provided by a diagram. For example, the type of U_2 and the owners of U_2 and U_3 are unspecified in the above diagram; a diagram model would fully specify those relations.

Diagram models consist of a set of analogical relations A and a set of label relations L over a universe U. Each member of A is a binary relation $E_s \times E_s$, with $E_s \subset U$ the set of diagram elements; each member of L is a relation $E_s \times E_l$, with $E_l \subset U$ the set of labels. Using the "displayed" format of [2, Section 1.3], we write these models as $\langle U, A, L, E_s, E_l \rangle$.

For the scene described by (1), the diagram elements are $\{V, U_1, U_2, U_3\}$. We choose the label relations $TYPE(u, l)$ and $OWNS(u, l)$, and the analogical relations $BES(u, v)$ (the opening u is next to the opening v) and $INHALL(u, v)$ (opening u is in hall v). The label set contains $\{Closet, Office, Ralph, Paul, Cyril\}$ and possibly other values. The choice of relations and elements is important in determining what information in the analogic structure is abstracted in the hybrid system; here, for example, whether an opening is to the right or left of another opening is apparent from the structure, but not in the models.

A key feature of analogical representations is their capacity to implicitly embody constraints that other representations must make explicit. For example, the map structures embed the following constraints:

- Each opening has at most 2 adjacent openings.
- Objects can have exactly 1 type.
- Individuals can own offices but not closets.
- At most one person can own a given office.

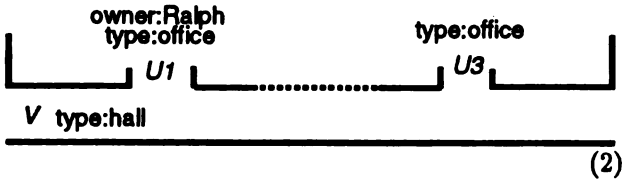
These *diagram constraints* can be built into the representation structures directly or into the operations that manipulate the structures, depending on the given implementation. For example, a bit-map representation of (1) would embed the first constraint directly through its spatial composition; the third constraint would most likely be enforced by operations that manipulate the structure. Either way, diagram constraints are necessarily reflected in diagram models. For instance, all diagram models for (1) can have only one type relation for a given diagram element, due to the second constraint above.

In diagram (1), all objects of relevance (the openings and the hall itself) have been noted and the analogical relations *BES* and *INHALL* are fully determined. Although there is type and ownership information missing, the *structure* of the diagram is complete. Not all diagrams share this completeness. When generating maps from perceptual input, noise or faulty sensors may both cause objects of interest to go undetected and leave analogical relations only partially determined. In such circumstances, we say that the diagram contains *structural uncertainty*. We formalize this notion as follows.

Definition 2.1 (Determined Relation) *A set of models M defined over a class of relations $R = \{r_1, \ldots r_m\}$* determines *a relation $r_i \in R$ iff every model in M agrees on the extension of r_i.*

Definition 2.2 (Structural Uncertainty) *A diagram S with models M_S is* structurally uncertain *iff some analogical relation of the models is undetermined.*

The following diagram constitutes a variation on the scene described by (1) in which there is structural uncertainty between U_1 and U_3. Here, both the *BES* and *INHALL* relations are undetermined. Dashed lines indicate regions of structural uncertainty:



(2)

As will be seen, our framework provides the means to apply sentential information about a diagram in order to both ascertain the composition of areas of structural uncertainty and flesh out the partial characterizations given by the diagram models for the relations in $L \cup A$.

2.2 Sentential Subsystem

The sentential subsystem employs a first-order language

$$\mathcal{L} = \langle \mathcal{P}_A, \mathcal{P}_L, E_s, E_l, \cdots \rangle$$

and a corresponding proof theory. For simplicity, we use the diagram elements E_s and labels E_l as standard names for themselves in $\mathcal{L}$. The predicates $\mathcal{P}_A$ are interpreted by the analogical relations of the diagram models, and $\mathcal{P}_L$ by the label relations. In addition, there may be other predicates and constants that have an indirect relation to the diagram – for example, the predicate $NBR(x, y)$ representing the office-neighbour relationship between two people. This predicate would be related to the diagram predicates $\mathcal{P}_A \cup \mathcal{P}_L$ by an axiom such as

$$\begin{aligned} \forall x, y.\ NBR(x, y) \equiv \\ \exists u, v.\ TYPE(u, Office) \wedge TYPE(v, Office) \\ \wedge OWNS(u, x) \wedge OWNS(v, y) \wedge BES(u, v)\ . \end{aligned} \tag{3}$$

Similarly, the predicate $RESIDES(x, h)$ representing the relationship of an individual x having an office in hallway h would be defined as

$$\begin{aligned} \forall x, h.\ RESIDES(x, h) \equiv \\ \exists u.\ INHALL(u, h) \wedge TYPE(u, Office) \\ \wedge\ OWNS(u, x)\ . \end{aligned} \tag{4}$$

We refer to axioms of this sort as *grounding* axioms.

As an example of the expression and use of sentential information relative to diagrams, consider the following statements:

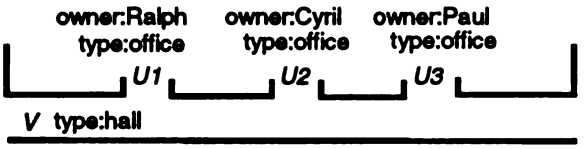
Paul and Cyril have offices in hall V.
Ralph and Paul are not neighbours.

Given the grounding axioms (3,4), these statements can be translated into the following formulas of $\mathcal{L}$:

$$\begin{aligned} RESIDES(Cyril, V) \wedge RESIDES(Paul, V) \\ \neg NBR(Ralph, Paul)\ . \end{aligned} \tag{5}$$

With respect to diagram (1), the first statement implies that U_2 and U_3 are offices, one each owned by Cyril and Paul. This conclusion follows since $\{U_1, U_2, U_3\}$ constitutes the set of all offices in V and *Ralph* is known to own U_1. Deduction of this result requires information that is implicit in the diagram's structure, namely that each office can be owned by only one individual. With the second statement, the

only possible configuration of the scene is:



(6)

Sentential information can also be used to reduce structural uncertainty in diagrams: given the sentences *Ralph and Cyril are neighbours* and *Cyril is Paul's only neighbour*, the diagram (6) follows from (2).

2.3 Integration Criteria

In order to determine whether a given integration method behaves in an appropriate fashion, it is necessary to provide a semantic account of the over-all hybrid system.

From a model-theoretic perspective, the merging of a diagram S with a set of sentences T that describe the diagram amounts to restricting the models of S to those that are *compatible* with T. Compatibility here means that the diagram model can be expanded to a model for T by providing interpretations for the predicate, function, and constant symbols of $\mathcal{L}$ that do not overlap the diagram model.

Definition 2.3 (Restricted Models) *Let T be a collection of sentences in $\mathcal{L}$ describing properties of a diagram S and let M_S be the models of S defined for the sublanguage $\langle \mathcal{P}_A, \mathcal{P}_L, E_s, E_l \rangle$ of $\mathcal{L}$. The* restriction of M_S relative to T, *written as $M_S(T)$, is the set of models $\langle U, A, L, E_s, E_l \rangle \in M_S$ for which some expansion $\langle U, A, L, E_s, E_l, \cdots \rangle$ is a model of T.*

The models in $M_S(T)$ characterize the total information content in the hybrid system for the domain modeled by the analogical structures. The challenge is to provide both *derivation rules* for determining formulas of $\mathcal{L}$ that are logically entailed by $M_S(T)$ and *update rules* for modifying S to eliminate diagram models not contained in $M_S(T)$.

In general, the analogical structures may have weaker representational capabilities than is required to capture the information content of $M_S(T)$. Consider the diagram (1) and the sentential theory

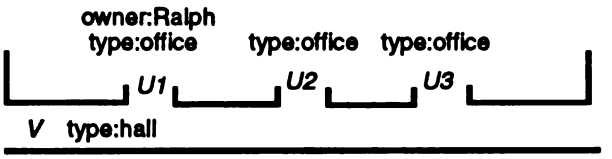
$$T_0 = \{RESIDES(Cyril, V), RESIDES(Paul, V)\} .$$

These two sources of information jointly imply that U_2 and U_3 are offices, one each owned by *Paul* and *Cyril*; however, it is undetermined as to who owns which one. Every model in $M_S(T_0)$ either has both $\langle U_2, Cyril \rangle$ and $\langle U_3, Paul \rangle$ or both $\langle U_2, Paul \rangle$ and $\langle U_3, Cyril \rangle$ in its *OWNS* relation. However, this information cannot be fully manifest in the diagram since it is not definite about who owns which office and the diagram does not admit disjunctive information about ownership.

Rather than seeking an analogical structure with the models $M_S(T)$, the best that can be attained is a structure that *adequately represent $M_S(T)$*:

Definition 2.4 (Representational Adequacy) *An analogical structure Q* adequately represents *a set of diagram models M iff $M \subseteq M_Q$ and there is no other diagram R such that $M \subseteq M_R$ and $M_R \subset M_Q$.*

For example, when S is the diagram (1) and T_0 is as defined above, the following diagram adequately represents $M_S(T_0)$:



(7)

This diagram extends (1) to include the information that U_2 is an office but does not include any new information about ownership.

Soundness and completeness for inference in our hybrid system can be defined using the concepts of restricted models and representational adequacy.

Definition 2.5 (Soundness) *A diagram update rule is* sound *iff for diagram S and theory T it generates only diagrams whose model set contains $M_S(T)$. A derivation rule is* sound *iff it generates only sentences whose model set contains $M_S(T)$.*

We will say that a collection of both diagram update and derivation rules is sound precisely when each of its members is sound.

Definition 2.6 (Completeness) *A set of derivation and update rules is* derivationally complete *for S and T iff any valid sentence of $M_S(T)$ can be derived by the sentential subsystem. The set is* diagrammatically complete *iff it can generate a diagram that adequately represent $M_S(T)$.*

3 The Inferential Calculus

The inferential calculus underlying our hybrid framework is defined relative to a class of domain-independent diagram operations. In this section, we describe both the inference rules and operations. We focus exclusively on diagrams without structural uncertainty; diagrams with structural uncertainty are considered in Section 4.

3.1 Diagram Operations

Two classes of diagram operations are required for our inference rules: *reflection* and *extraction* procedures.

Reflection procedures provide a means of inserting information into an analogical structure. For each label predicate $P(u,v)$ we require a reflection procedure INSERT.P(u,v) such that for $e_1, e_2 \in E_s \cup E_l$, the predicate $P(e_1,e_2)$ holds in all models of the diagram obtained by executing INSERT.P(e_1,e_2). That is, when applied to a diagram S with model set M_S, INSERT.P(e_1,e_2) yields a diagram S' with model set

$$M_{S'} = \{m \in M_S \mid m \models P(e_1,e_2)\} .$$

For diagrams without structural uncertainty (the focus of this section), insertion procedures for $\mathcal{P}_A$ are unnecessary.

Extraction procedures provide access to the contents of the analogical structure for use by the sentential subsystem. As noted above, whole-scale translation of the analogical structures into first-order sentences is infeasible. Instead, we wish to provide access to the information in the analogical structures on an *as needed* basis, whereby information is accessed as required for individual deduction steps rather than all at once. The two key types of information stored within diagrams are (1) analogical and label relationships for diagram elements, and (2) closure information about those relationships.

For each diagram predicate $P(u,v)$, we require an extraction procedure EVAL.P(u,v) for evaluating ground instances relative to the diagram S. These *evaluation* procedures provide the sentential reasoner with information about primitive relationships in the analogical structures. The procedure behaves as follows for $e_1, e_2 \in E_s \cup E_l$:

$$\text{EVAL.P}(e_1,e_2) = \begin{cases} \textit{true} & \text{if } M_S \models P(e_1,e_2) \\ \textit{false} & \text{if } M_S \models \neg P(e_1,e_2) \\ \textit{unknown} & \text{otherwise} \end{cases}$$

Closure information for a diagram S is generated by two classes of procedures. Let $P[x]$ represent an instance of a predicate in $\mathcal{P}_L \cup \mathcal{P}_A$ that contains the single variable x, such as $BES(x,U_1)$. (For simplicity, we restrict attention here to predicates containing only one variable.) With respect to the diagram S, the procedure CLOSURE$^+$.$P[x]$ generates the set of diagram elements that possibly satisfy $P[x]$ (called the *minimal superclosure*) while the procedure CLOSURE$^-$.$P[x]$ generates the set of elements that definitely satisfy $P[x]$ (the *maximal subclosure*).

Definition 3.1 (Closures) *Let $P[x]$ be a nonground instance of a predicate in $\mathcal{P}_L \cup \mathcal{P}_A$. The* minimal superclosure *of $P[x]$, denoted by* CLOSURE$^+$.$P[x]$, *and the* maximal subclosure *of $P[x]$, denoted by* CLOSURE$^-$.$P[x]$, *are defined with respect to a diagram S as follows:*

$$\text{CLOSURE}^+.P[x] = \{e \in E_l \cup E_s \mid m \models P[e] \text{ for some } m \in M_S\}$$

$$\text{CLOSURE}^-.P[x] = \{e \in E_l \cup E_s \mid M_S \models P[e]\}$$

The procedures CLOSURE$^+$.$P[x]$ and CLOSURE$^-$.$P[x]$ give minimal upper- and maximal lower-bounds, respectively, for the precise set of values that satisfy $P[x]$. This set is fixed for a given diagram only when the relation $P[x]$ is *determined* by the models of S (in the sense of Definition 2.1). In terms of the sentential language $\mathcal{L}$, determination of $P[x]$ is equivalent to the condition that for $m_1, m_2 \in M_S$:

$$\forall e \in E_s \cup E_l.\ m_1 \models P[e] \equiv m_2 \models P[e] . \quad (8)$$

When a predicate $P[x]$ is determined by a diagram, the maximal sub- and minimal superclosures are both equal to the exact closure. However, sub- and superclosures are useful sources of diagram information when the predicate is not determined, as will be made apparent in Section 3.2.3.

Note that since we are considering only diagrams without structural uncertainty in this section, all analogical predicates are necessarily determined.

3.2 Inference Rules

The inferential component of the integration framework consists of rules of *evaluation*, *domain enumeration* and *reflection*. Evaluation and domain enumeration utilize information from the diagram as provided by the extraction procedures to derive new sentences describing properties of the diagram. The reflection rule permits the insertion of sentential consequences derived from T into the diagrams using the reflection procedures.

In the definition of the inference rules, we use the notation α^c_b to represent the expression α with all occurrences of the expression b replaced by c.

3.2.1 Reflection

The reflection rule sanctions the transfer of information from the sentential to the analogical subsystem. Let $T \vdash \phi$ represent the deducibility of a sentence ϕ from a set of sentences T using the proof theory of the sentential subsystem.

Definition 3.2 (Reflection Rule) *Let T be a sentential theory and S an analogical structure. If $T \vdash R(t_1,\ldots,t_k)$ for $t_1,\ldots,t_k \in E_s \cup E_l$ and $R \in \mathcal{P}_A \cup \mathcal{P}_L$ then $R(t_1,\ldots,t_k)$ can be* reflected into *S by executing* INSERT.R$(t_1,\ldots,t_k)$.

3.2.2 Evaluation

The evaluation rule sanctions replacement of ground instances of a predicate $R \in \mathcal{P}_A \cup \mathcal{P}_L$ by either *true* or *false*, in accordance with the contents of the analogical structure. In the case where the relationship denoted by R is undetermined, the evaluation process has no effect.

Definition 3.3 (Evaluation Rule) *Let ϕ be a formula that contains an instance $R(t_1, \ldots, t_k)$ of a predicate $R \in \mathcal{P}_A \cup \mathcal{P}_L$. If* $\mathsf{EVAL.R}(t_1, \ldots, t_k) = \theta$ *where $\theta \in \{true, false\}$ then* evaluation of $R(t_1, \ldots, t_k)$ in ϕ *yields* $\phi^{\theta}_{R(t_1,\ldots,t_k)}$.

3.2.3 Domain Enumeration

The domain enumeration rules allow the elimination of quantifiers in certain cases through the introduction of an appropriate domain of values that covers the relevant instantiations of the quantified variable.

Consider the assertion

$$\exists u.\ BES(u, U_2) \wedge OWNS(u, Paul) \qquad (9)$$

relative to diagram (1). The interpretation of this formula is that the diagram element owned by Paul is located beside U_2. The conjunct $BES(u, U_2)$ limits the possibilities for this diagram element: according to (1), the element must be either U_1 or U_3. As such, the formula $OWNS(U_1, Paul) \vee OWNS(U_3, Paul)$ follows from (9). Similarly, the universally quantified formula

$$\forall u.\ INHALL(u, V) \supset TYPE(u, Office) \qquad (10)$$

can be viewed as a statement about the predicate $TYPE(u, Office)$, with $INHALL(u, V)$ serving as a filter on the set of relevant instantiations of the quantified variable. According to the diagram (1), the only values that satisfy $INHALL(u, V)$ are $\{U_1, U_2, U_3\}$ (*i.e.*, the exact closure of $INHALL(u, V)$ is $\{U_1, U_2, U_3\}$). Thus, the conjunction

$$\bigwedge_{d \in \{U_1, U_2, U_3\}} TYPE(d, Office)$$

is equivalent to (10) with respect to models for diagram (1).

We refer to the technique used above for applying closure information to eliminate quantifiers as *domain enumeration*. Domain enumeration does not apply to all predicate instances containing a quantified variable. The formula $\exists u.\ \neg BES(u, U_1) \wedge TYPE(u, Closet)$ illustrates this point. In this case, the exact closure for $BES(u, U_1)$ is not an appropriate restriction of the terms of $\mathcal{L}$; elimination of the existential quantifier using the exact closure would lead to unsound conclusions.

For existential quantifiers, the domain used in domain enumeration must include all bindings for which the embedded formula (*e.g.*, $BES(x, U_2) \wedge OWNS(x, Paul)$ in (9)) may have truth value *true*; this guarantees that all relevant instantiations of the quantified variable are covered. For universal quantifiers, the domain should exclude values for which the embedded formula is already determined to have truth value *true*. We call a predicate instance whose exact closure satisfies these conditions *focus expressions* for the given quantified formula. In essence, a focus expression prunes from consideration those bindings of a given quantified variable that do not provide useful information.

To formalize the concept of focus expressions, we introduce definitions for the *polarity* and *definiteness* of predicate instances in a formula.

Definition 3.4 (Polarity) *An instance of a predicate in a formula ϕ is called* positive *if the instance maps to an unnegated literal in the conjunctive normal form of ϕ and is called* negative *otherwise.*

Definition 3.5 (Definiteness) *An instance of a predicate in a formula ϕ is called* definite *if the instance maps to a literal in a clause of length one in the conjunctive normal form of ϕ and is called* indefinite *otherwise.*

We will combine the notions of polarity and definiteness, referring to individual instances as *negative indefinite* or *positive definite* as appropriate. The expression $INHALL(u, V)$ is a negative indefinite instance in (10) and a positive definite instance in

$$\exists u.\ INHALL(u, V) \wedge TYPE(u, Closet)\ .$$

Definition 3.6 (Focus Expression) *If ψ is a quantified formula (either $\forall z.\ \alpha$ or $\exists z.\ \alpha$) containing a predicate instance $P[z]$ then $P[z]$ is a* focus expression *for ψ iff either*

- *ψ has the form $\forall z.\ \alpha$ and the occurrence of $P[z]$ is negative indefinite,* OR
- *ψ has the form $\exists z.\ \alpha$ and the occurrence of $P[z]$ is positive definite.*

The domain enumeration rule is formally defined as follows.

Definition 3.7 (Domain Enumeration Rule) *If ψ is a quantified expression, either $\exists z.\ \alpha$ or $\forall z.\ \alpha$, containing a focus expression $\Phi[z]$ with maximal subclosure D^- and minimal superclosure D^+ then* domain enumeration *for ψ and $\Phi[z]$ yields:*

$$\bigvee_{d \in D^+} (\alpha^d_z) \qquad \text{if } \psi \text{ is } \exists z.\ \alpha$$

$$\bigwedge_{d \in D^-} (\alpha^d_z)^{true}_{\Phi[d]} \qquad \text{if } \psi \text{ is } \forall z.\ \alpha\ .$$

Note that when applying domain enumeration to a universally quantified formula $\forall z.\ \alpha[z]$, the embedded formula $\alpha[z]$ need not be fully retained. Instead, the simplification of $\alpha[z]$ in which the focus expression is replaced by *true* suffices, since the focus expression has truth value *true* for all terms in its maximal subclosure. For example, $INHALL(u, V) \supset TYPE(u, Office)$ can be reduced to the expression $TYPE(u, Office)$. Focus expressions must be retained for existentially quantified formulas: by definition the maximal superclosure may contain terms that are not in the exact closure (and hence do not satisfy the focus expression).

Domain enumeration could be extended to make use of nonatomic focus expressions. For example, the conjunction $BES(x, U_5) \wedge TYPE(x, Office)$ could serve as a focus expression in the formula

$$\forall x.\ BES(x, U_5) \wedge TYPE(x, Office) \supset OWNS(x, Eva)\ .$$

The intersection of the maximal subclosures for the individual conjuncts would serve as a more restricted (and hence more useful) domain for the universally quantified variable x. Similalry, the disjunction $BES(x, U_4) \vee BES(x, U_6)$ could be employed as a focus expression in

$$\exists x.\ (BES(x, U_4) \vee BES(x, U_6)) \wedge OWNS(x, Ann)\ .$$

The appropriate domain in this case would be the union of the minimal superclosures for each disjunct. Straightforward extensions of Definitions 3.6–3.7 support this generalization; we forego their technical statement in this paper.

3.3 Example

We illustrate the workings of our integration rules by applying them to the scenario presented in Section 2.2, namely diagram (1) with sentential theory

$$T_0 = \{\neg NBR(Ralph, Paul), RESIDES(Cyril, V), RESIDES(Paul, V)\}\ .$$

A derivation schematic, including both diagrams and sentences, is provided in Figure 1.

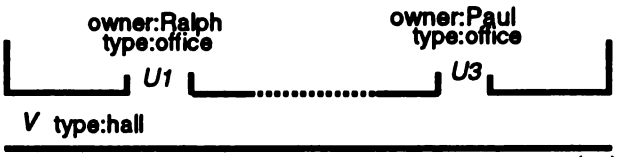
Consider first the given formula $RESIDES(Paul, V)$. Rewriting using definition (4) yields formula S2 in the figure. The predicate $INHALL(u, V)$ is a focus expression in S2 and its exact closure in diagram (a) is $\{U_1, U_2, U_3\}$. Domain enumeration using this focus expression yields formula S3. Diagram (a) contains the information that U_1 and U_3 are offices, thus $TYPE(U_1, Office)$ and $TYPE(U_3, Office)$ in S3 can be replaced by *true* using the evaluation rule. In addition, since the diagram indicates that *Ralph* owns U_1, evaluation can be used to rewrite $OWNS(Paul, U_1)$ to *false*. This evaluation step exploits the diagram constraint that ownership is unique. The evaluations combined with tautological simplification produce formula S4.

Expansion of the given fact $\neg NBR(Ralph, Paul)$ using definition (3) gives formula S6 in the figure. This formula contains the focus expression $OWNS(Ralph, x)$ whose exact closure in diagram (a) is $\{U_1\}$; domain enumeration is applied to produce formula S7. Evaluation of the expression $TYPE(U_1, Office)$ with respect to diagram (a) leads to formula S8, which contains the focus expression $BES(U_1, y)$ whose exact closure is $\{U_2\}$. Domain enumeration for $BES(U_1, y)$ yields formula S9, which along with S4 entails $OWNS(Paul, U_3)$. Application of the reflection rule to this atom yields the second diagram (b).

The formula S12 is obtained from the given formula $RESIDES(Cyril, V)$ by applying the same steps used from S1 to S4 above. The ownership of U_3 was undetermined in the original diagram; however, $OWNS(Cyril, U_3)$ is necessarily *false* since the new diagram (b) indicates that the owner of U_3 is *Paul*. Evaluation can be applied to the formula S12 using diagram (b) to derive S13. Note that this last deduction could not be made from S12 and the sentence $OWNS(Paul, U_3)$ alone; again we need the diagram constraint that ownership is unique. Finally, the contents of this last formula can be reflected to produce the diagram (c), which adequately represents $M_S(T_0)$.

4 Structural Uncertainty

Consider the diagram



(11)

containing a region of structural uncertainty between elements U_1 and U_3. This diagram has models in which there are zero, one, two, *etc*, diagram elements in the uncertain area. Without further information, there is no way to determine which of these models corresponds to the actual situation that the diagram is intended to represent.

Accounting for structural uncertainty requires a slight generalization of the inferential calculus presented in Section 3. First of all, reflection operators must be provided for the analogical predicates $\mathcal{P}_A$ so that diagrams can be modified to incorporate new analogical relations determined by the sentential subsystem. The definitions of the remaining diagram operations and the various inference rules remain unaltered but the definition of minimal superclosure requires elaboration. Because the minimal superclosure of a predicate must include all possible values for which the predicate holds, it is necessary to consider whether elements inserted in structurally indeterminate regions could satisfy the given predicate. In contrast, the definition of maximal subclosure is not affected by structural uncertainty since diagram elements that do not appear in all models are not included in the maximal

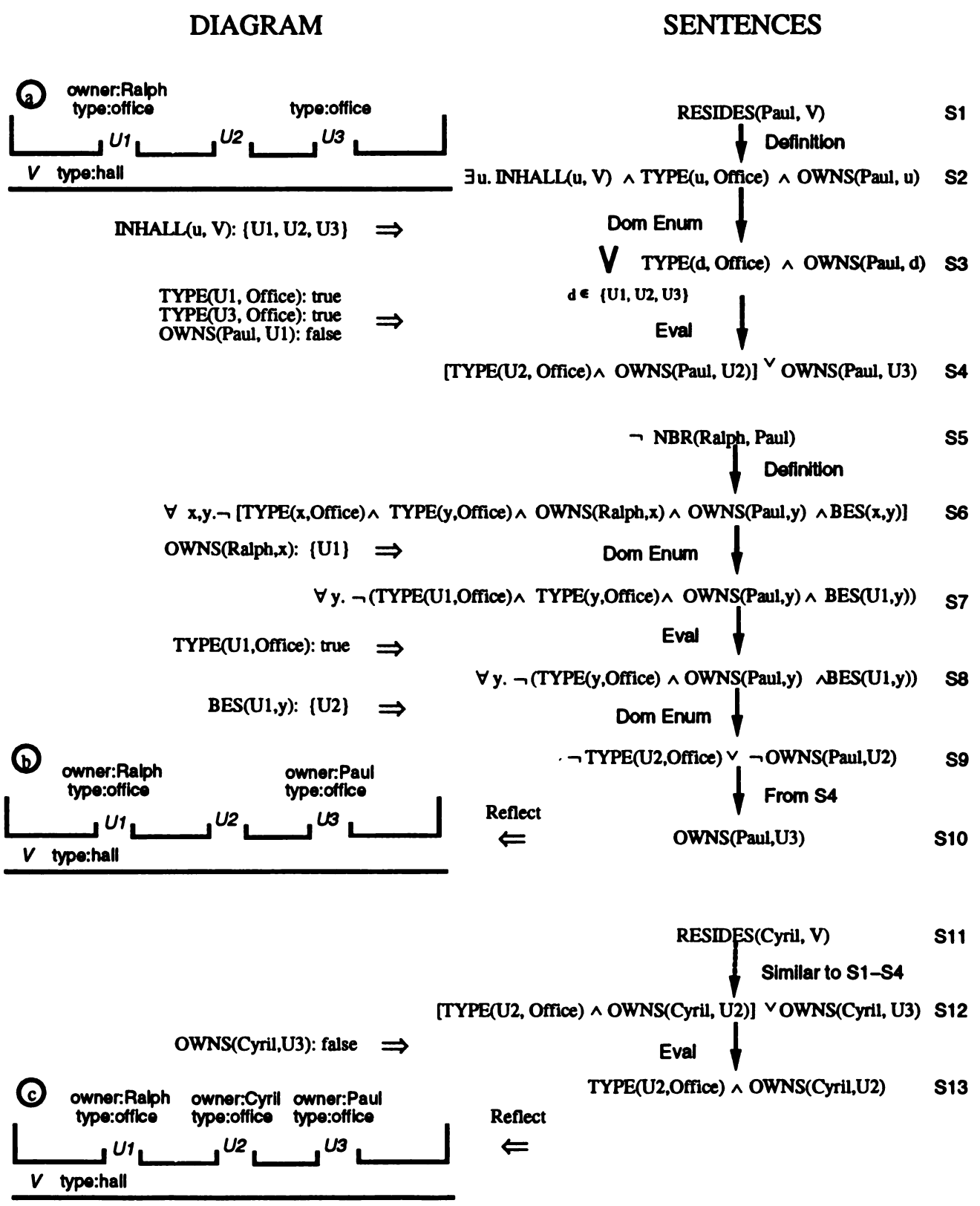


Figure 1: An Example Derivation

subclosure.

4.1 Minimal Superclosures with Introduced Names

To account for structural uncertainty in diagrams, we exploit a technique of the natural deduction calculus for dealing with existential elimination. With this calculus, the existential quantifier of a formula $\exists x.\phi[x]$ can be eliminated by introducing a *new* individual constant c for x, yielding $\phi[c]$. The justification for the introduction is that, since c does not appear elsewhere in the proof, it can refer to an arbitrary individual.

We employ the same principle in formulating minimal superclosures for diagrams containing structural uncertainty. Consider diagram (11) relative to the sentence $\exists x.\ BES(U_1, x)$. It could be that U_3 is next to U_1, or that there is an intervening element I_1 situated to the right of U_1. The minimal superclosure must take both of these cases into account, yielding the set $\{U_3, I_1\}$.

Employing names for hypothesized individuals introduces a complication to the diagrams, since we have heretofore assumed that all elements were "standardized apart," receiving different names if and only if they were distinct. This is not the case with hypothesized individuals; for example, if we were later to perform domain enumeration on the sentence $\exists y.\ BES(U_3, y)$, introducing the name I_2, it could be the case that both I_1 and I_2 refer to the same individual. To account for naming and identity, we assume that the diagram keeps track of introduced names and their possible referents.[1]

When the exact closure of an analogical predicate $P[x]$ is *determined* by the diagram models (*i.e.*, condition (8) is satisfied), the minimal superclosure reduces to the exact closure. Otherwise, the minimal superclosure consists of those diagram elements that satisfy the predicate in any diagram model, along with an introduced name for a hypothesized element. Even though multiple elements can appear in regions of uncertainty and there may be more than one such region in a diagram, only one introduced element is required for the minimal superclosure. Restriction to one such element is possible because the purpose of domain enumeration is to identify a solitary element satisfying the matrix of the existentially quantified formula.

Definition 4.1 (Minimal Superclosure for $\mathcal{P}_A$) *Let $P[x]$ represent an instance of a predicate in $\mathcal{P}_A$ that contains the single variable x. The* minimal superclosure *of $P[x]$, denoted by* $\mathsf{CLOSURE}^+.P[x]$, *is defined for a diagram S as*

$$\{e \in E_l \cup E_s \mid M_S \models P[e]\}$$

when $P[x]$ is determined by M_S, otherwise

$$\{e \in E_l \cup E_s \mid m \models P[e] \text{ for some } m \in M_S\} \cup \{I_k\}$$

where I_k is an introduced name.

The minimal superclosure for $BES(U_1, x)$ relative to diagram (11) is $\{U_3, I_1\}$, where I_1 is an introduced name.

The new definition of minimal superclosure is used as before in domain enumeration except that when an element name I_k is introduced for a minimal superclosure, the diagram is modified to include inequalities between I_k and all current diagram elements. The referent of an introduced I_k may be determined by future sentential reasoning steps, possibly leading to the insertion of a new diagram element *via* the reflection rule. Such a situation arises in the example presented below.

4.2 Example: Structural Uncertainty

Consider diagram (11) and the theory

$$T_1 = \{NBR(Ralph, Cyril),\ NBR(Paul, Cyril)\}\ .$$

Every model in $M_S(T_1)$ has exactly one element between U_1 and U_3, with this element being labeled as the office of *Cyril*. We show how a new diagram that reflects this information can be generated using the integration calculus.

Applying the same steps used to derive S8 from $\neg NBR(Ralph, Paul)$ in Figure 1, we generate the following pair of formulas from T_1:

$$\exists x.\ BES(U_1, x) \wedge TYPE(x, Office) \wedge OWNS(Cyril, x) \tag{12}$$

$$\exists x.\ BES(U_3, x) \wedge TYPE(x, Office) \wedge OWNS(Cyril, x)\ . \tag{13}$$

As noted above, the minimal superclosure for $BES(U_1, x)$ relative to diagram (11) is $\{U_3, I_1\}$. Domain enumeration for the focus expression $BES(U_1, x)$ in (12) yields

$$\begin{array}{c} BES(U_1, U_3) \wedge TYPE(U_3, Office) \wedge OWNS(Cyril, U_3) \\ \vee \\ BES(U_1, I_1) \wedge TYPE(I_1, Office) \wedge OWNS(Cyril, I_1) \end{array} \tag{14}$$

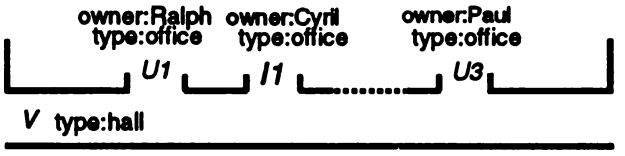
along with the naming constraints $U_3 \neq I_1$ and $U_1 \neq I_1$. Evaluation of $OWNS(Cyril, U_3)$ with respect to the diagram returns *false* (since ownership is unique), thus (14) simplifies to

$$BES(U_1, I_1) \wedge TYPE(I_1, Office) \wedge OWNS(Cyril, I_1)\ . \tag{15}$$

The conjuncts in this formula can be reflected to pro-

[1] In other words, diagram operations must track equalities and inequalities for introduced names.

duce the new diagram



(16)

Here, a new diagram element has been created to serve as the referent of the introduced name I_1.

In diagram (16), the expression $BES(x, U_3)$ has minimal superclosure $\{I_1, I_2\}$, for some introduced name I_2. Domain enumeration for $BES(x, U_3)$ in (13) gives

$$BES(U_3, I_1) \wedge TYPE(I_1, Office) \wedge OWNS(Cyril, I_1)$$
$$\vee$$
$$BES(U_3, I_2) \wedge TYPE(I_2, Office) \wedge OWNS(Cyril, I_2) \quad (17)$$

along with the inequalities $I_2 \neq U_1$, $I_2 \neq U_3$, and $I_2 \neq I_1$.

Since the current diagram indicates that *Cyril* owns I_1 (and cannot own I_2 since $I_2 \neq I_1$), the evaluation rule can be applied to eliminate the second disjunct of (17). Further evaluations lead to the formula $BES(U_3, I_1)$, thus establishing that the introduced name I_2 does not refer to a realizable diagram element. Reflection of this last atom yields the diagram (6), which adequately represents $M_S(T_1)$.

4.3 A Troublesome Example

The new definition of minimal superclosure does not always lead to an adequate representation of the restricted class of diagram models. Suppose that we relate sentences (12,13) to the following diagram D:

(18)

The models $M_D(T_1)$ are adequately represented by a diagram similar to (6) but with regions of uncertainty to the left of U_1 and to the right of U_3. As we will show, our integration calculus cannot produce this diagram.

The minimal superclosure for $BES(U_1, x)$ is again $\{U_3, I_1\}$ as was the case in the previous example (provided we reuse I_1 as the introduced name) and we can similarly derive the formula (15) from (12). However, since I_1 could be located on either side of U_1 we cannot insert a new element into the diagram as the referent of I_1. This inability to situate I_1 leads to a different minimal superclosure for $BES(U_3, x)$, namely $\{U_1, I_2\}$ (again we reuse the same introduced name from the previous example). Domain enumeration for $BES(x, U_3)$ in (13) now gives

$$BES(U_3, U_1) \wedge TYPE(U_1, Office) \wedge OWNS(Cyril, U_1)$$
$$\vee$$
$$BES(U_3, I_2) \wedge TYPE(I_2, Office) \wedge OWNS(Cyril, I_2)$$

with the inequalities $I_2 \neq U_1$ and $I_2 \neq U_3$. Note that in this case, the inequality $I_2 \neq I_1$ is not added since I_1 does not refer to a current diagram element.

At this point, no modifications to the diagram are possible, and no further derivations by the sentential subsystem lead to any reflections back to the diagram. While the atoms $BES(U_1, I_1)$, $BES(U_3, I_2)$, $OWNS(Cyril, I_1)$ and $OWNS(Cyril, I_2)$ are all derivable, they have no effect on the diagram individually. In combination though, they constrain $I_1 = I_2$ to be the unique office situated between U_1 and U_2. Generating a diagram that adequately represents $M_D(T_1)$ requires reasoning by cases about the possible locations of introduced elements and is beyond the scope of the integration calculus presented in this paper.

5 Properties of the Framework

The integration framework satisfies the following properties.

Proposition 5.1 (Soundness) *Reflection, evaluation and domain enumeration are sound.*

An inference rule that derives a formula ψ from a given formula ϕ is *equivalence-preserving* with respect to a class of models M when $M \models \phi \equiv \psi$.

Proposition 5.2 (Equivalence) *Domain enumeration using exact closures for a diagram S is an equivalence-preserving inference rule with respect to the models of S.*

The proofs of the propositions are quite straightforward; we defer them to a longer version of the paper.

The integration rules are neither derivationally nor diagrammatically complete. The central problem is that the rules focus on properties of individuals and their relationships with other individuals, failing to account for embedded diagram constraints. Diagram constraints are certainly made use of at times. In going from S3 to S4 in Figure 1, the unique ownership constraint made it possible to conclude that $OWNS(Paul, U_1)$ had truth-value *false*, given that $OWNS(Ralph, U_1)$ held in diagram (a). However, it is impossible to directly reason with the diagram constraints.

In the remainder of this section we briefly consider the issue of completeness for propositional sentential theories. We note that the discussion is also of relevance to those first-order theories that can be reduced to

propositional theories through appropriate use of the domain enumeration rule.

Derivational completeness demands the derivability in the sentential subsystem of any sentence valid in $M_S(T)$. Suppose we pick a propositionally-complete refutation strategy for the sentential subsystem. Is the resulting system complete? The answer is "no." Consider the following set of statements:

$$\begin{array}{rcl} OWNS(Paul, U_1) & \vee & OWNS(Paul, U_2) \\ OWNS(Ralph, U_1) & \vee & OWNS(Ralph, U_2) \\ OWNS(Cyril, U_1) & \vee & OWNS(Cyril, U_2) \end{array}$$

Given the embedded property of unique ownership, these sentences are inconsistent with respect to *any* diagram in our class of maps. Even so, it is not alway possible to derive the empty clause. In particular, no refutation is possible given a variation of diagram (1) in which all ownership information has been removed. Because the uniqueness constraint on ownership is embedded in the representations and operations of the analogical structures, the integration rules provide no means of relating this constraint to the sentences above.

Derivational completeness can be attained by extending the evaluation rule to sets of literals. Define:

$$\mathsf{EVAL}^*(l_1, \cdots l_n) = \begin{cases} \textit{inconsis} & \text{if } M_S \models \neg(l_1 \wedge \cdots \wedge l_n) \\ \textit{unknown} & \text{otherwise} \end{cases}$$

Using this evaluation procedure, we can apply total narrow theory resolution [25] as a refutation-complete derivational procedure. This procedure is a variant of hyperresolution in which a set of literals, one from each clause of the resolution, are tested for consistency against the diagram; if they are inconsistent, the result of the resolution is a clause consisting of a disjunction of the remainders of each resolved clause.[2]

Diagram completeness is generally harder to achieve than derivational completeness because it requires the sentential subsystem to be complete for atomic consequence-finding. Consider the theory:

$$T = \{OWNS(Paul, U_3) \vee OWNS(Cyril, U_3)\}$$

relative to diagram (1). Given the embedded diagram constraint that ownership applies only to offices, it is possible to derive $TYPE(U_3, Office)$ by refutation. But $TYPE(U_3, Office)$ cannot be derived as a consequence of the diagram and T. Arriving at this conclusion requires a form of reasoning by cases that our framework does not currently support. As noted above, reasoning by cases is also required to overcome the form of diagram incompleteness that arose in Section 4.3. This capability presents an interesting avenue for further research.

[2] Although we can achieve derivational completeness using theory resolution in a refutation system, in practice we might not want to use theory resolution directly, since it is a multiple-clause inference rule. It would be more efficient to consider a variation on the Davis-Putnam method in which branches are closed when they contain a set of atoms that are inconsistent according to the diagram.

6 Summary

We have described a domain-independent formal framework for integrating sentential and analogical representations. We illustrated the workings of the framework for the application of reasoning sententially to extend the information content of maps, both with and without structural uncertainty. The integration rules of the framework are sound as well as derivationally complete in the propositional case.

The integration framework has been implemented on top of the KLAUS automated deduction system [26] using the method of *universal attachment* [14, 15] to formulate the integration rules. The system has been successfully applied to problems involving reasoning with maps, including the examples presented in this paper. Much work remains to be done on control issues for the implementation. In particular, the ordering of domain enumerations can greatly impact the efficiency of reasoning.

There has been a resurgence of interest in computational models for diagrammatic/visual reasoning during the past few years. Most similar in nature to our research is the work on Hyperproof [1]. The Hyperproof system combines sentential reasoning with diagrammatic representations of a chessboard containing blocks. In Hyperproof, much of the complexity underlying the integration of diagrammatic and sentential information is implicit in the system; in contrast, our research has sought to provide a domain-independent inferential framework in which all aspects of integration are made explicit. A second difference relates to control: the inference rules presented here automatically combine sentential and diagrammatic reasoning, while Hyperproof requires user interaction to guide the reasoning process.

The work of [17] presents a computational model for reasoning with diagrammatic representations but emphasizes the emulation of human reasoning about visualization. *Computational imagery* [19] defines a representational framework for reasoning with both visual and spatial information but does not address the connection to deductive inference.

Our work also overlaps to a certain extent with research in the hybrid reasoning community [4, 3]. Other than our specialization of hybrid reasoning to analogical representations, the main difference between the material presented there and the hybrid framework presented here is the latter's emphasis on reflecting derived information back into analogical structures.

Acknowledgements

This research was supported by the Office of Naval Research under Contract N00014-89-C-0095.

References

[1] J. Barwise and J. Etchemendy. Visual information and valid reasoning. In W. Zimmerman, editor, *Visualization in Mathematics*. Mathematical Association of America, Washington, DC, 1990.

[2] C. Chang and H. J. Keisler. *Model Theory*. Elsevier Press, New York, 1977.

[3] A. M. Frisch, editor. *Proceedings of the AAAI 1991 Fall Symposium on Principles of Hybrid Reasoning*, 1991.

[4] A. M. Frisch and R. B. Scherl. A bibliography on hybrid reasoning. *AI Magazine*, 11(5), 1991.

[5] B. V. Funt. Problem-solving with diagrammatic representations. *Artificial Intelligence*, 13, 1980.

[6] G. W. Furnas. Formal models for imaginal deduction. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 662–669. Lawrence Erlbaum, 1990.

[7] F. Gardin and B. Meltzer. Analogical representations of naive physics. *Artificial Intelligence*, 38:139–159, 1989.

[8] H. Gelernter. Realization of a geometry-theorem proving machine. In E. A. Feigenbaum and J. Feldman, editors, *Computers and Thought*. McGraw-Hill, New York, 1963.

[9] P. J. Hayes. Some problems and non-problems in representation theory. In *Proceedings of the AISB Summer Conference*, pages 63–79, University of Sussex, 1974.

[10] P. N. Johnson-Laird. Mental models in cognitive science. *Cognitive Science*, 4:71–115, 1980.

[11] S. M. Kosslyn. *Image and Mind*. Harvard University Press, Cambridge, MA, 1980.

[12] J. H. Larkin and H. A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:65–99, 1987.

[13] K. L. Myers. *Universal Attachment: An Integration Method for Logic Hybrids*. PhD thesis, Stanford University, 1991.

[14] K. L. Myers. Universal attachment: An integration method for logic hybrids. In J. A. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*. Morgan Kaufmann, 1991.

[15] K. L. Myers. Hybrid reasoning using universal attachment. *Artificial Intelligence*, 1992. To appear.

[16] K. L. Myers and K. Konolige. Integrating analogical and sentential reasoning for perception. In *Proceedings of the AAAI Spring Symposium on Reasoning with Diagrammatic Representations*, 1992.

[17] N. Narayanan and B. Chandrasekaran. Reasoning visually about spatial interactions. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 1991.

[18] E. P. Novak, Jr. and W. C. Bulko. Understanding natural language with diagrams. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 465–470, 1990.

[19] D. Papadias and J. I. Glasgow. A knowledge representation scheme for computational imagery. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*. Lawrence Erlbaum Associates, 1991.

[20] S.-J. Shin. An information-theoretic analysis of valid reasoning with Venn diagrams. In J. Barwise, J. M. Gawron, G. Plotkin, and S. Tutiya, editors, *Situation Theory and its Applications*, volume 2. 1991.

[21] S.-J. Shin. *Valid Reasoning and Visual Representation*. PhD thesis, Dept. of Philosophy, Stanford University, 1991.

[22] A. Sloman. Interactions between philosophy and AI. *Artificial Intelligence*, 2, 1971.

[23] A. Sloman. Afterthoughts on analogical representation. In *Proceedings of Theoretical Issues in Natural Language Processing*, 1975.

[24] K. Stenning and J. Oberlander. Spatial containment and set membership. In J. Barnden and K. Holyoak, editors, *Analogical Connections*. 1992.

[25] M. E. Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1(4), 1985.

[26] M. E. Stickel. The KLAUS automated deduction system. In *Proceedings of the Ninth International Conference on Automated Deduction*, 1988.

Order of Magnitude Reasoning using Logarithms

P. Pandurang Nayak
Knowledge Systems Laboratory
Stanford University
701 Welch Road, Bldg. C, Palo Alto, CA 94304
nayak@cs.stanford.edu

Abstract

Converting complex equations into simpler, more tractable equations usually involves approximation. Approximation is usually done by identifying and removing insignificant terms, while retaining significant ones. The significance of a term can be determined by order of magnitude reasoning. In this paper we describe NAPIER, an implemented order of magnitude reasoning system. NAPIER defines the order of magnitude of a quantity on a logarithmic scale, and uses a set of rules to propagate orders of magnitudes through equations. A novel feature of NAPIER is the way it handles non-linear simultaneous equations, using linear programming in conjunction with backtracking. We show that order of magnitude reasoning in NAPIER is, in general, intractable and then discuss an approximate reasoning technique that allow it to run fast in practice. Some of NAPIER's inference rules are heuristic, and we estimate the error introduced by their use.

1 INTRODUCTION

Mathematical models are pervasive in science and engineering. Both analytical and numerical techniques have been used to solve the equations resulting from such models. Analytical methods are applicable only to restricted classes of equations (e.g., linear systems). To apply analytical techniques to more complex classes of equations, scientists and engineers have had to find ways of approximating the equations to convert them into a simpler form.

With the advent of fast digital computers, the classes of equations that can be solved by numerical (rather than analytical) methods have grown considerably. However, numerical methods are not a panacea; they have their limitations too. For example, solving systems of non-linear equations can be time consuming, and a good initial guess is required to ensure convergence to a solution. Hence, scientists and engineers still strive to identify appropriate approximations so that the resulting equations are as simple as possible. In addition, numerical methods are sometimes inapplicable. For example, during conceptual design, exact numerical values for exogenous quantities are usually unavailable because most of the details of the design are unspecified. Under such circumstances, an engineer needs to fall back on analytical methods, thereby highlighting the role of appropriate approximations.

The approximation process usually involves identifying and removing insignificant terms, while retaining only the significant ones. Consider the following example, previously discussed in [Bennett, 1987; Raiman, 1991], from the domain of acid-base chemistry. An important task in this domain is to find the concentration of H^+ ions in a solution. The concentration of ions in solution depends on the dynamic equilibrium resulting from competing chemical reactions. Consider dissolving an acid, AH, in water. The two reversible reactions that occur, corresponding to the ionization of AH and H_2O, are shown in Figure 1.

$$\begin{aligned} AH &\rightleftharpoons H^+ + A^- \\ H_2O &\rightleftharpoons H^+ + OH^- \end{aligned}$$

Figure 1: Ionization reactions that occur on dissolving AH in water

The equilibrium concentrations of the three ions (H^+, OH^-, A^-) and the acid (AH) are determined by the equations shown in Figure 2. Square brackets denote concentrations; C_a is the initial concentration of the acid; K_w is the ion product of water; and K_a is the ionization constant of the acid.

As has been pointed out in [Bennett, 1987; Raiman, 1991], solving this set of equations analytically for $[H^+]$ results in a cubic equation which is difficult to solve. In fact, in problems involving *polyprotic* acids, i.e., acids that can yield more than one H^+ ion, the

Charge balance:	$[H^+] = [A^-] + [OH^-]$	(1)
Mass balance:	$C_a = [A^-] + [AH]$	(2)
Acid ionization equilibrium:	$K_a[AH] = [A^-][H^+]$	(3)
Water ionization equilibrium:	$K_w = [OH^-][H^+]$	(4)

Figure 2: Equilibrium equations for the ionization reactions.

closed form solution for $[H^+]$ can involve equations of degree five or higher, making the solution significantly harder.

An alternative to the above approach is to approximate the equations, and hence simplify them. For example, a chemist might guess that the acid is strong, so that $[A^-] \gg [OH^-]$ and $[A^-] \gg [AH]$. This justifies reducing the first equation to $[H^+] = [A^-]$ and the second equation to $C_a = [A^-]$, leading to a straightforward solution.

The reasoning following the assumptions that $[A^-] \gg [OH^-]$ and $[A^-] \gg [AH]$ is very nicely formalized in [Raiman, 1991]. But how are these assumptions justified? In [Bennett, 1987], Bennett suggests that such assumptions are justified by *domain specific* inference rules. In this paper we present a *domain-independent* method for justifying such assumptions. In particular, we define the *order of magnitude* of a quantity on a logarithmic scale. We show how the order of magnitude of exogenous quantities like C_a, K_w, and K_a can be propagated through a set of equations like Equations 1–4 to compute orders of magnitudes of the remaining quantities like $[H^+]$, $[OH^-]$, $[A^-]$, and $[AH]$. The computed orders of magnitude of $[A^-]$, $[OH^-]$, and $[AH]$ can be used to justify the above assumptions and simplify the equations.

The reasoning technique described here has been implemented in a program called NAPIER.[1] NAPIER has been successfully tested on multiple examples in the domain of electromechanical devices, with the number of equations per example ranging from about 25 to a little over 150.

In the next section, we define the order of magnitude of a quantity and present rules for propagating orders of magnitude. We show that these rules can be used to infer orders of magnitudes of quantities related by a set of non-linear simultaneous equations. Next, we show that, in general, order of magnitude reasoning is intractable, and then present an approximate reasoning technique that make it efficient in practice. Since some of the order of magnitude reasoning rules are heuristic, we next estimate the error introduced by their use. We conclude with a discussion of related work.

[1] John Napier (1550–1617), a Scottish nobleman, is credited with the first discovery of logarithms.

2 ORDER OF MAGNITUDE REASONING IN NAPIER

Order of magnitude reasoning in NAPIER is a form of interval reasoning. The *order of magnitude* of a quantity q (denoted $om(q)$) is defined as follows:

$$om(q) = \lfloor \log_b |q| \rfloor \quad (5)$$

where the base, b, of the logarithm is chosen to be the smallest number that can be considered to be "much larger" than 1. The choice of b is clearly domain and task dependent. In this paper we assume that $b = 10$. Note that the order of magnitude of a quantity, q, is independent of its sign, and hence $om(q) = om(-q)$. In what follows, we assume that the signs of all quantities have been determined, to the extent possible, prior to any reasoning about orders of magnitude using standard constraint satisfaction techniques.[2]

2.1 INFERENCE RULES IN NAPIER

Given the orders of magnitude of q_1 and q_2, NAPIER computes bounds on the orders of magnitude of arithmetic expressions involving q_1 and q_2, using the rules shown in Figure 3. The rules for $(q_1 + q_2)$ and $(q_1 - q_2)$ assume that q_1 and q_2 have the same sign so that the magnitudes of q_1 and q_2 are actually being added or subtracted, respectively. The rule for $(q_1 \pm q_2)$ is applicable to a sum or difference of q_1 and q_2 when the sign of at least one of q_1 and q_2 is unknown.

The rules for $(q_1 * q_2)$ and (q_1/q_2) (rules 1 and 2) follow directly from Equation 5 and the rules of interval arithmetic [Moore, 1979]. For example, if $om(q_1) = n_1$ and $om(q_2) = n_2$, it follows that $b^{n_1} \leq |q_1| < b^{n_1+1}$ and $b^{n_2} \leq |q_2| < b^{n_2+1}$. Using interval arithmetic, we get $b^{n_1+n_2} \leq |q_1 * q_2| < b^{n_1+n_2+2}$, and hence $n_1 + n_2 \leq om(q_1 * q_2) \leq n_1 + n_2 + 1$.

Like rules 1 and 2, rules 3a and 4a are also based on Equation 5 and interval arithmetic. Note, however, that these rules predict larger intervals for $(q_1 + q_2)$ and $(q_1 - q_2)$ than interval arithmetic predicts under the same restrictions on q_1 and q_2. For example, if

[2] This assumption is unnecessarily strong. For example, if a and b are positive, constraint satisfaction alone is unable to deduce the sign of $a-b$. However, if $om(a) > om(b)$, then $a - b$ can be deduced to be positive.

1. $om(q_1) + om(q_2) \leq om(q_1 * q_2) \leq om(q_1) + om(q_2) + 1$
2. $om(q_1) - om(q_2) - 1 \leq om(q_1/q_2) \leq om(q_1) - om(q_2)$
3. a) $om(q_1) \leq om(q_1 + q_2) \leq om(q_1) + 1$ if $om(q_1) = om(q_2)$
   b) $om(q_1 + q_2) = om(q_1)$ if $om(q_1) > om(q_2)$
   c) $om(q_1 + q_2) = om(q_2)$ if $om(q_1) < om(q_2)$
4. a) $om(q_1 - q_2) \leq om(q_1)$ if $om(q_1) = om(q_2)$
   b) $om(q_1 - q_2) = om(q_1)$ if $om(q_1) > om(q_2)$
   c) $om(q_1 - q_2) = om(q_2)$ if $om(q_1) < om(q_2)$
5. a) $om(q_1 \pm q_2) \leq om(q_1) + 1$ if $om(q_1) = om(q_2)$
   b) $om(q_1 \pm q_2) = om(q_1)$ if $om(q_1) > om(q_2)$
   c) $om(q_1 \pm q_2) = om(q_2)$ if $om(q_1) < om(q_2)$

Figure 3: Rules for order of magnitude reasoning

$om(q_1) = om(q_2) = n$, then interval arithmetic predicts that $(q_1 + q_2)$ is bounded by $2b^n$ and $2b^{n+1}$, while NAPIER predicts $n \leq om(q_1 + q_2) \leq n + 1$, which is equivalent to saying that $(q_1 + q_2)$ lies between b^n and b^{n+2}. This larger interval is a consequence of NAPIER being able to represent only intervals whose end points are integer powers of the chosen base. Further note that rules 3a and 4a are correct only if the base is greater than 2. This is reasonable given our heuristic for selecting the base (viz., 2 is unlikely to be considered to be "much larger" than 1).

Unlike the rules discussed thus far, rules 3b, 3c, 4b, and 4c are not guaranteed to be correct, but are heuristic rules. They are all based on the intuition that adding or subtracting a "small" quantity from a "large" quantity does not significantly affect the larger quantity. Since the base in Equation 5 is chosen as the smallest number that can be considered to be "much larger" than 1, the above intuition justifies these rules; the order of magnitude of a quantity is not affected by adding or subtracting quantities of a smaller order of magnitude. The inclusion of these heuristic order of magnitude rules differentiates NAPIER from standard interval reasoners. In Section 5, we estimate the error introduced by the use of these heuristic rules.

Finally, rule set 5 merely encompasses both rule sets 3 and 4. It is used to infer the order of magnitude of a sum or difference of two quantities when the signs of at least one of the two quantities is not known. To determine the order of magnitude of a sum or difference of two quantities, NAPIER selects the appropriate rule set from rule sets 3, 4, and 5, depending on the operation (sum or difference) and the signs of the two quantities. For example, consider the equation $q_3 = q_1 + q_2$. If q_1 and q_2 have the same sign, then rule set 3 is used to infer $om(q_3)$; if q_1 and q_2 have opposite signs, then rule set 4 is used to infer $om(q_3)$, since the magnitude of q_3 is really the difference of the magnitudes of q_1 and q_2; and if the signs of at least one of q_1 and q_2 is unknown, then rule set 5 is used to infer $om(q_3)$.

2.2 SET OF SIMULTANEOUS EQUATIONS

Until now, we have focussed exclusively on how NAPIER uses a single equation to propagate orders of magnitudes, i.e., how $om(q_1 op\ q_2)$ is computed from $om(q_1)$ and $om(q_2)$. However, the rules in Figure 3 can also be used to compute orders of magnitudes of quantities related by a set of (possibly non-linear) simultaneous equations. NAPIER uses these rules to convert a set of simultaneous equations into a set of constraints, where each constraint is a disjunction of a set of linear inequalities. Each equation in the set of simultaneous equations contributes a constraint as follows:

1. Product and quotient terms contribute a single set of linear inequalities according to rules 1 and 2, respectively. For example, $q_3 = q_1 * q_2$ contributes the following set:

$$\{om(q_1) + om(q_2) \leq om(q_3),\ om(q_3) \leq om(q_1) + om(q_2) + 1\}$$

2. Sum and difference terms contribute a disjunction of three sets of linear inequalities, using rule sets 3, 4, or 5, as applicable. Each disjunct corresponds to one of the rules (a, b, or c) in the applicable rule set. For example, assuming that q_1 and q_2 have the same sign, the equation $q_3 = q_1 - q_2$ contributes the following disjunction:[3]

$$\{om(q_3) \leq om(q_1), om(q_1) = om(q_2)\}$$
$$\vee$$
$$\{om(q_3) = om(q_1), om(q_1) \geq om(q_2) + 1\}$$
$$\vee$$
$$\{om(q_3) = om(q_2), om(q_1) \leq om(q_2) - 1\}$$

corresponding to rules 4a, 4b, and 4c, respectively.

NAPIER uses this set of constraints to compute bounds on the orders of magnitudes of the quantities. Since all

[3]Since the order of magnitudes are integral, $om(q_1) > om(q_2)$ is equivalent to $om(q_1) \geq om(q_2) + 1$.

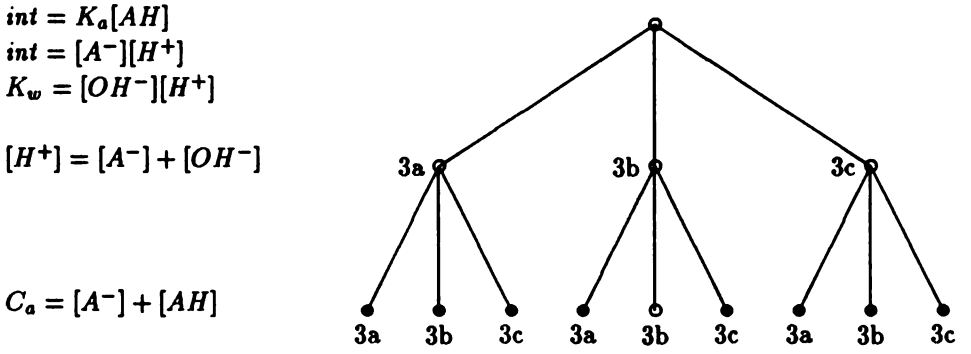


Figure 4: A backtrack tree.

the inequalities in the constraints are linear inequalities, NAPIER uses linear programming [Hillier and Lieberman, 1980], in conjunction with backtracking, to compute order of magnitude bounds. Backtracking is necessary to handle the disjunctions. We describe this algorithm next.

2.3 BACKTRACKING ALGORITHM

Let E denote the set of simultaneous equations being processed. NAPIER's backtracking procedure is best visualized as a depth-first traversal of a backtrack tree. Each level in the tree (except the root level) corresponds to one of the sum or difference terms in E. The root level corresponds to all the product and quotient terms in E. Each internal node has three children, corresponding to the three disjuncts in the constraint contributed by the sum or difference term at the level of the node's children. Each node in the tree has an associated set of linear inequalities defined as follows:

1. The set of inequalities at the root node consists of the union of the sets of inequalities contributed by each product and quotient term in E.
2. The set of inequalities at each non-root node consists of the union of (a) the inequalities at the node's parent; and (b) the inequalities in the disjunct associated with that node.

Starting at the root node, NAPIER traverses the backtrack tree in a depth-first manner. At each node it checks the consistency of the inequalities at that node. If the set is inconsistent, it immediately backtracks to the node's parent. If the set is consistent and it is a non-leaf node, it continues its depth-first traversal. If the set is consistent and it is a leaf node, it uses the inequalities to find the maximum and minimum values of the order of magnitude of each quantity. The bounds computed at each of the consistent leaf nodes are combined so that the lower bound of each quantity is the least lower bound and the upper bound is the greatest upper bound.

Since the inequalities at each node are linear, NAPIER uses the Simplex linear programming algorithm [Hillier and Lieberman, 1980; Press *et al.*, 1989] to check their consistency, and to compute the order of magnitude bounds at leaf nodes. However, from Equation 5 it follows that the order of magnitude of a quantity is integral. Hence, instead of using linear programming, NAPIER should use integer programming [Hillier and Lieberman, 1980]. Unfortunately, integer programming is known to be intractable [Karp, 1972], which leads to severe restrictions on the number of equations and the size of the backtrack tree that can be handled. Hence, to avoid such restrictions, NAPIER uses linear programming.

It is important to note that, while bounds computed by linear programming are not guaranteed to be tight[4], they are guaranteed to be correct: upper bounds will be greater than or equal to integer programming upper bounds, and lower bounds will be less than or equal integer programming lower bounds. In addition, we have found that, in practice, linear programming bounds are usually integral, in which case there is no loss of solution quality.

2.4 EXAMPLE

We now illustrate the above procedure using Equations 1–4. Let us assume that the exogenous orders of magnitude are as follows: $om(K_w) = -14$, $om(K_a) = -2$, $om(C_a) = -5$. This corresponds to a moderately strong solution of a strong acid. The backtrack tree resulting from these equations is shown in Figure 4. The equations associated with each level are shown on the left of the tree. Note that Equation 3 had to be split into two product expressions, with the introduction of an intermediate variable *int*. The rules (and hence the disjuncts) associated with each non-root node are displayed near each node. Nodes that are filled in are

[4] A bound b_1, interpreted as an interval, is said to be *tight* with respect to a bound b_2 if b_1 and b_2 are identical. b_1 is *looser* than b_2 if b_1 contains b_2.

the inconsistent nodes. For example, the left most leaf node can be seen to be inconsistent using the following line of reasoning. Applying rule 3a to Equations 1 and 2, we get

$$om([OH^-]) = om([A^-]) = om([AH])$$
$$om([A^-]) \leq om(C_a) \leq om([A^-]) + 1$$
$$om([A^-]) \leq om([H^+]) \leq om([A^-]) + 1$$

Since $om(C_a) = -5$, it follows that the least value of $om([A^-])$ is -6. Hence the least values of $om([OH^-])$ and $om([H^+])$ are also -6, and hence the least value of $om([OH^-][H^+])$ is -12. But rule 1 applied to Equation 4 requires that:

$$om([OH^-][H^+]) = om(K_w) = -14$$

which leads to a contradiction.

Of course, NAPIER doesn't need the above line of reasoning to infer inconsistencies; it reaches the same conclusion using linear programming.

The only consistent set of inequalities at the leaf nodes is the middle most leaf node, corresponding to assuming that $om([A^-]) > om([OH^-])$ and $om([A^-]) > om([AH])$. The quantity bounds calculated at this node are as follows:[5]

$$\begin{aligned} om([H^+]) &= -5 \\ om([OH^-]) &= (-10, -9) \\ om([AH]) &= (-9, -7) \\ om([A^-]) &= -5 \end{aligned}$$

Since $[A^-]$ is at least two orders of magnitude greater than $[AH]$, and at least four orders of magnitude greater than $[OH^-]$, a chemist is justified in making the assumptions that $[A^-] \gg [OH^-]$ and $[A^-] \gg [AH]$. These assumptions can then be used to simplify the equations, as discussed earlier.

A slight variation of the above example illustrates the importance of having such justifications. Suppose that, instead of having $om(C_a) = -5$, we had $om(C_a) = -8$. This corresponds to a weak solution of the same strong acid. Using this new value for $om(C_a)$, NAPIER predicts the following bounds on the orders of magnitude:

$$\begin{aligned} om([H^+]) &= -7 \\ om([OH^-]) &= (-8, -7) \\ om([AH]) &= (-14, -12) \\ om([A^-]) &= -8 \end{aligned}$$

These values justify the assumption that $[A^-] \gg [AH]$, but the other assumption, $[A^-] \gg [OH^-]$, is seen to be completely unjustified. This means that only Equation 2 can be simplified. Hence, NAPIER is a useful tool in justifying the order of magnitude assumptions that scientists and engineers make in simplifying equations.

[5] $om(q) = (l, u)$ represents the fact that $l \leq om(q) \leq u$

In addition to its role in justifying order of magnitude assumptions, NAPIER's predictions can also be used directly. For example, if all the chemist is interested in is the approximate pH of the solution[6], then NAPIER's predictions can be used directly: in the first case, the pH is between 5 and 4; in the second case, the pH is between 7 and 6. Note that NAPIER was able to make these predictions using approximate values of C_a, K_w, and K_a. This feature makes it particularly useful during conceptual design.

3 ORDER OF MAGNITUDE REASONING IS INTRACTABLE

The backtracking algorithm described in the previous section, generates a tree whose worst case size is exponential in the number of sum and difference expressions. In this section we show that order of magnitude reasoning using the rules in Figure 3 is intractable, even if orders of magnitude are not required to be integral. This means that NAPIER can do little better than generate a backtrack tree whose worst case size is exponential.

We start by defining the decision problem corresponding to finding the maximum order of magnitude of a quantity:

Definition 1 *(*ORDER OF MAGNITUDE REASONING*) Let E be a set of equations, and let V be the set of quantities used in E. Let $X \subseteq V$ be the set of exogenous quantities, with known orders of magnitude. Let $q \in V$ be a quantity and let B be an integer. Let $s : V \rightarrow \{+, -, unknown\}$ be a function that assigns signs to the quantities in V. (Quantities with unknown signs are assigned "unknown.") Assuming that the order of magnitude of a quantity is not required to be integral, is the maximum value of $om(q)$, derived using the rules in Figure 3 on the set E, greater than or equal to B?*

We now state the following theorem without detailed proof:

Theorem 1 *The* ORDER OF MAGNITUDE REASONING *problem is NP-complete.*[7]

The proof of this theorem is based on a reduction from an arbitrary instance of 3SAT. Briefly, the reduction introduces a quantity for each literal in the instance of 3SAT. Equations are added to ensure that quantities corresponding to complementary literals have the property that the order of magnitude of one of them must be 0 and the order of magnitude of the other one must be 1. The mapping between truth assignments

[6] The pH of a solution is defined to be $-\log_{10}[H^+]$.

[7] See [Garey and Johnson, 1979] for a comprehensive introduction to the theory of intractability.

Example number	Number of equations	Number of +/− terms	Time (sec) on an Explorer II	
			All equations	With causal ordering
1	28	11	2733	2.0
2	31	11	2435	1.0
3	45	14	–	2.9
4	60	24	–	2.7
5	80	25	–	37.2
6	110	32	–	35.9
7	111	32	–	94.6
8	119	35	–	20.4
9	145	43	–	45.2
10	163	50	–	21.0

Table 1: NAPIER's run times with and without causal ordering.

and orders of magnitudes is straightforward: a literal is true if and only if the corresponding quantity's order of magnitude is 1. Additional equations involving the above quantities and the special quantity q are then introduced, and the bound B is defined to ensure that the maximum value of $om(q)$ is greater than or equal to B if and only if all the clauses are satisfied. See [Nayak, 1992] for details.

Assuming $P \neq NP$, Theorem 1 tells us that, in the worst case, NAPIER will have to generate a backtrack tree whose size is exponential in the number of sum and difference terms. Unfortunately, the exponential blow up does occur in practice. We have used NAPIER in the domain of electromechanical devices, as part of the automated model selection system described in [Nayak *et al.*, 1992]. Table 1 summarizes NAPIER's performance on models of ten different devices (see [Nayak, 1992] for a description of the devices).

The second column in this table shows the total number of equations in each example, while the third column shows the the total number of sum and difference terms. The fourth column shows the time it took NAPIER to run its backtracking algorithm on the complete set of equations.[8] NAPIER was given a maximum of one hour to solve each example; a "–" entry in column four denotes that NAPIER could not solve the example in an hour. As is clear from the table, only the two smallest examples could be solved in under an hour, each taking over 40 minutes. Hence, NAPIER appears to be quite impractical, except for the smallest examples. To make it practical, we now develop an approximate reasoning scheme for NAPIER that trades off accuracy for speed.

4 APPROXIMATION ALGORITHMS IN NAPIER

The backtrack tree developed by NAPIER is, in the worst case, exponential in the number of sum and difference terms in the set of equations under consideration. Hence, to make NAPIER practically useful, it is important to decrease the number of sum and difference terms that are handled at any one time. We now discuss a method for doing this, based on a *dependency ordering* of the equations.

4.1 ORDERING THE EQUATIONS

The dependency ordering of equations that we consider is the *causal ordering*, described in [Iwasaki and Simon, 1986]. The causal ordering specifies the order in which equations are to be solved, and identifies minimal sets of equations that *must* be solved simultaneously. The causal ordering can be viewed as a directed acyclic graph. Each node in the graph consists of a set of equations that must be solved simultaneously. There is an edge from node n_1 to node n_2 if the equations at n_2 use a quantity whose value is determined by the equations at n_1.

NAPIER processes the equation sets in the order specified by the causal ordering: equation sets earlier in the ordering are processed first. NAPIER bounds the orders of magnitudes of the quantities used in an equation set, and uses these bounds as exogenous bounds for equation sets later in the ordering.

The use of the above dependency ordering has a significant computational advantage. A large set of equations, with many sum and difference terms, can often be broken down into many small sets of equations, with each equation set having very few sum and difference terms. Hence, NAPIER can process each equation set in the dependency ordering very fast. Column five in Table 1 shows the time it took NAPIER to solve the ten examples using causal ordering. It takes NAPIER from a few seconds to under two minutes to solve each of these examples, showing that causal ordering has made NAPIER practical for large sets of equations.

[8] The fifth column will be discussed in the next section.

Example #	Δ_{max}	Example #	Δ_{max}
1	11	6	7
2	11	7	4
3	9	8	5
4	7	9	6
5	7	10	6

Table 2: Maximum value of Δ for each example.

4.2 LOSS OF ACCURACY

The drawback of using the dependency ordering is that global constraints can be lost, leading to excessively loose bounds on the orders of magnitudes. Consider, for example, the set $\{y_1 = x_1 * x_2, y_2 = x_3/y_1, y_3 = y_1 * y_2\}$, and let x_1, x_2, and x_3 be exogenous with orders of magnitude 0. The dependency ordering generated from this set of equations is:

$$\begin{array}{ccc} & \{y_2 = x_3/y_1\} & \\ \nearrow & & \searrow \\ \{y_1 = x_1 * x_2\} & \longrightarrow & \{y_3 = y_1 * y_2\} \end{array}$$

Using this dependency ordering, NAPIER computes the order of magnitude of y_3 as follows: from the first equation it computes $om(y_1)$ to be between 0 and 1; from the second equation, and the calculated bound on $om(y_1)$, it computes $om(y_2)$ to be between -2 and 0; and from the third equation and the calculated bounds on $om(y_1)$ and $om(y_2)$, it computes $om(y_3)$ to be between -2 and 2. However, if all three equations were considered simultaneously, NAPIER computes $om(y_3)$ to be between -1 and 1.

The reason for the looser bound in the first case stems from not enforcing some global constraints. For example, the lower bound of $om(y_3)$ can be -2 only when $om(y_1) = 0$ and $om(y_2) = -2$. However, when $om(y_1)$ is 0, the second equation dictates that the lowest that $om(y_2)$ can be is -1. This fact is lost when the third equation is processed by itself.

More generally, the above problem occurs when a quantity, like y_3, depends on two or more quantities, like y_1 and y_2, whose values have been determined by equations that are earlier in the causal ordering. In using these previously determined values, NAPIER disregards any additional constraints that might hold between those values. Hence, bounds computed based on these values may not be as tight as possible.

NAPIER can partially address this problem by combining adjacent sets of equations in the dependency ordering. This allows more equations to be handled simultaneously, so that more global constraints can be incorporated. However, combining adjacent sets of equations can lead to an increase in the number of sum and difference terms that must be handled simultaneously. Hence, adjacent sets are combined only when the number of sum and difference terms in the resulting set does not increase beyond a threshold (call this threshold Δ).

Combining adjacent sets of equations, as described above, also allows us to partially empirically evaluate the effect of causal ordering on accuracy. We ran NAPIER a number of times on each of our examples, using increasing values of Δ, allowing a maximum of 15 minutes per run. Table 2 shows the maximum value of Δ used for each example. We then compared the bounds that were computed without combining adjacent sets with the bounds that were computed with the maximum setting of Δ. Interestingly, we found that there was *no* loss of accuracy—the bounds computed with and without combining adjacent sets were identical.

To understand the reason for this somewhat surprising result, we now analyze the source of the additional constraints on previously determined values. Let us assume that $om(p_3)$ is computed using previously computed values of $om(p_1)$ and $om(p_2)$. Additional constraints on the values of $om(p_1)$ and $om(p_2)$ stem from one of two sources: (a) $om(p_1)$ and $om(p_2)$ are determined simultaneously; and (b) the value of $om(p_1)$ is used in computing the value of $om(p_2)$, i.e., the values of one of these quantities depends on the value of the other. Point (a) manifests itself as a node in the causal ordering which contains more than one equation. Point (b) manifests itself as multiple paths between two nodes in the causal ordering.

Hence, if the causal ordering, viewed as a graph, satisfies the following two properties:

1. each node contains exactly one equation; and
2. there is at most one path between any two nodes;

then we can show that there will be *no* additional constraints between previously determined values. Hence, there is no loss of accuracy in using the causal ordering.

Table 3 shows how closely the causal orderings generated from our examples match the above two properties. The second and third columns of this table show the maximum and average number of equations per node, respectively. One can see that, in all cases, the average number of equations per node is very close to 1. The fourth column shows the minimum number of edges that must be removed from the causal ordering to ensure that there is at most one path between any

Example number	Equations per node		# of extra edges
	Maximum	Average	
1	7	1.27	1
2	7	1.24	0
3	7	1.15	1
4	1	1.00	0
5	12	1.29	1
6	18	1.29	2
7	17	1.26	6
8	9	1.25	2
9	18	1.21	3
10	16	1.10	0

Table 3: Properties of the causal ordering graph

two nodes. One can see that, in most cases these numbers are very small. Hence, the above analysis provides us with some insight into the reasons underlying the fact that, in our examples, the bounds computed with and without combining adjacent sets are identical.

5 ERROR ESTIMATION

In this section, we estimate the error introduced by the use of the heuristic rules introduced in Section 2.1. We then analyze some alternate order of magnitude rules that seem intuitively plausible, and show that these rules introduce unacceptably large errors. The analysis is done using probability theory and is based on interpreting each quantity as a *random variable*.[9]

5.1 ESTIMATING THE ERROR OF HEURISTIC RULES

We start by analyzing rule 3b. Let Q, Q_1, and Q_2 be quantities such that $Q = Q_1 + Q_2$. Let f_{Q_1} and f_{Q_2} be the probability density functions of Q_1 and Q_2, respectively, and let f_{Q_1,Q_2} be their joint probability density function. (Briefly, $f_{Q_1}(q_1)$ is the probability that Q_1 lies between q_1 and q_1+dq_1, and $f_{Q_1,Q_2}(q_1,q_2)$ is the probability that Q_1 lies between q_1 and q_1+dq_1, and Q_2 lies between q_2 and q_2+dq_2.) Since $Q = Q_1 + Q_2$, it follows that the probability that Q lies between l and u, for any values l and u, is:

$$Prob\{l \leq Q < u\} = \int_{-\infty}^{\infty} \int_{l-q_1}^{u-q_1} f_{Q_1,Q_2}(q_1,q_2) dq_2 dq_1 \quad (6)$$

Let us now assume that $om(Q_1) = n_1$ and $om(Q_2) = n_2$, with $n_1 > n_2$. Under these conditions, rule 3b states that $om(Q) = n_1$, i.e., $b^{n_1} \leq Q < b^{n_1+1}$. To estimate the error, ϵ(Rule 3b), in rule 3b, we must calculate the probability that Q lies outside the region from b^{n_1} to b^{n_1+1}:

$$\begin{aligned} &\epsilon(\text{Rule 3b}) \\ &= 1 - Prob\{b^{n_1} \leq Q < b^{n_1+1}\} \\ &= 1 - \int_{-\infty}^{\infty} \int_{b^{n_1}-q_1}^{b^{n_1+1}-q_1} f_{Q_1,Q_2}(q_1,q_2) dq_2 dq_1 \quad (7) \end{aligned}$$

To evaluate this integral, we make the following assumptions:

Assumption 1: Q_1 and Q_2 are independent random variables. Hence, the joint probablity density of Q_1 and Q_2 is just the product of the individual probablity densities:

$$f_{Q_1,Q_2}(q_1,q_2) = f_{Q_1}(q_1) f_{Q_2}(q_2) \quad (8)$$

Assumption 2: Q_1 and Q_2 are uniformly distributed on the intervals $[b^{n_1}, b^{n_1+1})$ and $[b^{n_2}, b^{n_2+1})$, respectively:

$$f_{Q_1}(q_1) = \begin{cases} \frac{1}{b^{n_1+1}-b^{n_1}} & \text{if } b^{n_1} \leq q_1 < b^{n_1+1} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{Q_2}(q_2) = \begin{cases} \frac{1}{b^{n_2+1}-b^{n_2}} & \text{if } b^{n_2} \leq q_2 < b^{n_2+1} \\ 0 & \text{otherwise} \end{cases}$$

Using these assumptions, we get the following result (see [Nayak, 1992] for details):

$$\epsilon(\text{Rule 3b}) = \frac{b+1}{2b^{n_1-n_2}(b-1)} \quad (9)$$

Hence, under Assumptions 1 and 2, the error in rule 3b is maximum when $(n_1 - n_2)$ is minimum, i.e., $(n_1 - n_2) = 1$, which occurs when quantities of consecutive orders of magnitude are being added. When $b = 10$, the maximum error is 6.11%. The error in rule 4b can also be shown to be $(b+1)/2b^{n_1-n_2}(b-1)$ in a similar way.

5.2 ALTERNATE ORDER OF MAGNITUDE RULES

The above error estimation techniques can also be used to analyze the alternate order of magnitude reasoning

[9] See [Davenport, 1970] for an introduction to probability theory and random variables.

rules shown in Figure 5. We have selected these rules because they seem intuitively very appealing. However, while these rules may appear intuitively appealing, they are also unacceptably error-prone.

1') $om(q_1 * q_2) = om(q_1) + om(q_2)$

2') $om(q_1/q_2) = om(q_1) - om(q_2)$

3a') $om(q_1 + q_2) = om(q_1)$ if $om(q_1) = om(q_2)$

Figure 5: Alternate rules for order of magnitude reasoning.

In particular, we can use the techniques in the previous section to show that:

$$\epsilon(\text{Rule } 1') = 1 - (b \ln b - b + 1)/(b-1)^2 \quad (10)$$

$$\epsilon(\text{Rule } 2') = 1/2 \quad (11)$$

$$\epsilon(\text{Rule } 3a') = 1 - (b-2)^2/2(b-1)^2 \quad (12)$$

Substituting $b = 10$ into the above equations tells us that the error in rule 1' is 82.68%, the error in rule 2' is 50%, and the error in rule 3a' is 60.49%. We believe that these errors are unacceptably large, and hence have chosen not to include these rules in NAPIER.

6 RELATED WORK

Order of magnitude reasoning has been widely studied in AI. Murthy [Murthy, 1988] was the first to propose the use of a logarithmic scale for the order of magnitude of a quantity. In that paper, he also provides rules of inference to infer new orders of magnitude from old ones. Some of these rules are similar to ours. For example, he includes rules 3b, 3c, 4b, and 4c. However, instead of 1, he proposes the rule $om(q_1 * q_2) = om(q_1) + om(q_2)$ (which is rule 1'), and instead of rule 3a, he proposes the rule $om(q_1 + q_2) = om(q_1)$ when $om(q_1) = om(q_2)$ (which is rule 3a'). As we saw in Section 5.2, the estimated error in these rules is too large, and hence we have chosen not to include them in NAPIER. Unlike our work, Murthy provides no analysis of how his inference rules can be used to find the order of magnitudes of quantities related by sets of simultaneous equations. In addition, we also analyze the complexity of order of magnitude inference, and present an approximate reasoning technique that works well in practice.

Raiman [Raiman, 1991; Raiman, 1986] explores the foundations of symbolic order of magnitude reasoning. He defines a variety of order of magnitude scales, such as *Close* and *Comparable*, built out of the basic order of magnitude granularities, *Small* and *Rough*. He introduces ESTIMATES, a system to solve order of magnitude equations. The primary difference between NAPIER and ESTIMATES is one of emphasis: NAPIER can be viewed as providing justifications for making order of magnitude assumptions; ESTIMATES can be viewed as a formalization of the use of such order of magnitude assumptions to symbolically manipulate and simplify equations.

Order of magnitude reasoning in the O(M) formalism [Mavrovouniotis and Stephanopolous, 1987] uses a parameter e to represent the largest quantity that can be considered to be "much smaller" than 1. This is analogous to the parameter b in NAPIER (i.e., $b = 1/e$). However, there are a number of differences between O(M) and NAPIER. First, the O(M) formalism is based on order of magnitude relations *between* quantities. Hence, it works best when equations involve only *links* (links are ratios of quantities). NAPIER, on the other hand, is based on the order of magnitudes of the quantities themselves, and hence works with any algebraic equations. This is advantageous because it is not always possible to convert equations into equations involving only links. Second, O(M) requires equations to be converted into *assignments*, which allow a new relation or range to be inferred from already known relations. This is a serious restriction since equations can be converted to assignments only in the absence of simultaneous equations. As we have seen, NAPIER does not have this restriction.

NAPIER is also related to interval reasoning discussed in [Moore, 1979; Simmons, 1986; Sacks, 1987]. NAPIER can be viewed as interval reasoning in which the end points of the interval are restricted to a particular set of points of the form b^n, with specified base b, and any integer n. The drawback of this restriction is that under certain conditions, compared to interval reasoning, the bounds inferred by NAPIER are unnecessarily loose (e.g., see the discussion of rule 3a in Section 2.1). The advantage of this restriction is that, unlike traditional interval reasoners, NAPIER is able to use sets of non-linear simultaneous equations to infer quantity bounds. In addition, the ability to simultaneously process all the equations in a set allows NAPIER to exploit global constraints to compute tighter bounds (see Section 4). Another distinguishing characteristic of NAPIER, which classifies it as an order of magnitude reasoning system rather than just an interval reasoner, is the use of heuristic rules (e.g, rule 3b).

7 CONCLUSIONS

In this paper we described an implemented order of magnitude reasoning system called NAPIER. NAPIER defines the order of magnitude of a quantity on a logarithmic scale and uses a set of rules to propagate order of magnitudes through equations. A novel feature of NAPIER is its handling of non-linear simultaneous equations. Since the order of magnitude reasoning rules are all disjunctions of linear inequalities, NAPIER is able to use linear programming, in conjunction with backtracking, to find bounds on the order of

magnitudes of quantities related by sets of non-linear simultaneous equations.

We also showed that order of magnitude reasoning using NAPIER's rules is intractable. Hence, NAPIER uses an approximate reasoning technique, based on causal ordering, leading to a practically useful system. This approximate reasoning technique trades off accuracy for speed, though in practice there does not appear to be any loss of accuracy.

Some of NAPIER's rules are heuristic rules, and we have estimated the error introduced by the use of these rules. We have also shown that intuitively appealing alternate heuristic rules lead to large estimated errors.

NAPIER has been extensively used in an automated model selection system described in [Nayak *et al.*, 1992]. We believe that NAPIER will find wide spread applications in different aspects of engineering and scientific problem solving.

Acknowledgements

I would like to thank Richard Fikes, Pat Hayes, and Leo Joskowicz for useful discussions on order of magnitude reasoning, and for comments on earlier drafts of this paper. Thanks also to the two anonymous reviewers, whose detailed comments helped improve the paper. Pandurang Nayak was supported by an IBM Graduate Technical Fellowship. Additional support for this research was provided by the Defense Advanced Research Projects Agency under NASA Grant NAG 2-581 (under ARPA order number 6822), by NASA under NASA Grant NCC 2-537, and by IBM under agreement number 14780042.

References

[Bennett, 1987] Bennett, Scott W. 1987. Approximation in mathematical domains. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Los Altos, CA. International Joint Conferences on Artificial Intelligence, Inc., Morgan Kaufmann Publishers, Inc. 239–241.

[Davenport, 1970] Davenport, Wilbur B. Jr. 1970. *Probability and Random Processes.* McGraw-Hill Book Company.

[Garey and Johnson, 1979] Garey, Michael R. and Johnson, David S. 1979. *Computers and Intractability.* W. H. Freeman and Company.

[Hillier and Lieberman, 1980] Hillier, Fredrick S. and Lieberman, Gerald J. 1980. *Introduction to Operations Reesearch.* Holden-Day, Inc., third edition.

[Iwasaki and Simon, 1986] Iwasaki, Yumi and Simon, Herbert A. 1986. Causality in device behavior. *Artificial Intelligence* 29:3–32.

[Karp, 1972] Karp, Richard M. 1972. Reducibility among combinatorial problems. In Miller, R. E. and Thatcher, J. W., editors 1972, *Complexity of Computer Computations.* Plenum Press, New York. 85–103.

[Mavrovouniotis and Stephanopolous, 1987] Mavrovouniotis, M. and Stephanopolous, G. 1987. Reasoning with orders of magnitude and approximate relations. In *Proceedings of the Sixth National Conference on Artificial Intelligence.* American Association for Artificial Intelligence.

[Moore, 1979] Moore, Ramon E. 1979. *Methods and Applications of Interval Analysis.* SIAM Studies in Applied Mathematics. SIAM, Philadelphia.

[Murthy, 1988] Murthy, Seshashayee S. 1988. Qualitative reasoning at multiple resolutions. In *Proceedings of the Seventh National Conference on Artificial Intelligence.* American Association for Artificial Intelligence. 296–300.

[Nayak *et al.*, 1992] Nayak, P. Pandurang; Joskowicz, Leo; and Addanki, Sanjaya 1992. Automated model selection using context-dependent behaviors. In *Proceedings of the Tenth National Conference on Artificial Intelligence.* American Association for Artificial Intelligence.

[Nayak, 1992] Nayak, P. Pandurang 1992. *Automated Modeling of Physical Systems.* Ph.D. Dissertation, Stanford University, Department of Computer Science, Stanford, CA.

[Press *et al.*, 1989] Press, William H.; Flannery, Brian P.; Teukolsky, Saul A.; and Vetterling, William T. 1989. *Numerical Recipes in Pascal: The Art of Scientific Computing.* Cambridge University Press.

[Raiman, 1986] Raiman, Olivier 1986. Order of magnitude reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence.* American Association for Artificial Intelligence. 100–104.

[Raiman, 1991] Raiman, Olivier 1991. Order of magnitude reasoning. *Artificial Intelligence* 51:11–38.

[Sacks, 1987] Sacks, Elisha 1987. Hierarchical reasoning about inequalities. In *Proceedings of the Sixth National Conference on Artificial Intelligence.* American Association for Artificial Intelligence, Morgan Kaufmann Publishers, Inc. 649–654.

[Simmons, 1986] Simmons, Reid 1986. "Commonsense" arithmetic reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence.* American Association for Artificial Intelligence. 118–124.

III.

Issues in Multi-Agent Environments

Semantics for knowledge and communication

Adam J. Grove
Department of Computer Science,
Stanford University, CA 94305.
grove@cs.stanford.edu

Abstract

Given a system of agents that interact by exchanging messages, we address the issue of understanding the *content* of the messages sent, and how this content interacts with agents' knowledge. We investigate very simple multi-agent systems, and describe various logics and semantics for ascribing content to messages. The main theme of this paper is that, as we consider more general classes of system, several subtleties arise and more expressive logics are required. The most expressive scheme we consider in detail uses two indexical terms, "I" and "You", which, roughly speaking, refer to the sender and recipient(s) respectively.

1 Introduction

Consider a collection of autonomous agents that interact by exchanging messages. What do these agents actually say to each other? At one level, messages are simply bit strings sent along wires, radio-wave modulations, visible signals, and so on. Of course, at this low level, the state of the agents themselves consists of values stored in memory, sensor readings, and so on. More abstract characterizations of the agents are also possible. In artificial intelligence, we believe that it can be useful to speak in terms of various mental attitudes, such as agents' knowledge, beliefs, desires, and intentions. Analogously, we would like a high-level description of the meaning and purpose of messages used in communication.

Of course, there is a vast amount of work in this general area already; for instance, speech act theory (e.g., [CP79]). However, existing work makes most sense at a rather abstract level, where concepts such as belief and intention are essentially taken as primitives. However, what happens if we want to deal with actual systems, i.e., real hardware and software? The problem is that there is, as yet, no comprehensive theory telling us how to ascribe mental concepts to to agents. How can we determine what a given machine "desires"?

This general goal—relating abstract concepts to real systems—is a worthwhile one. After all, people clearly find it useful to reason about simple systems in terms of the abstractions: My alarm clock *tells* me, each morning, that "It is six o'clock"; I know, but do not care, about the more detailed description in terms of electronic timers and loud harsh noises. It is often argued that this way of viewing the world is relevant to artificial intelligence (e.g., McCarthy in [McC79]). Another compelling reason for understanding how the abstractions can be grounded in reality is the following. Many people believe that it will be easier to design complex systems in terms of such intuitive notions such as beliefs and goals. But the final product of thinking like this is not an actual system, but rather, a high level specification. It is therefore important to understand when a real system is faithful to such an abstract description; without this, we cannot determine whether a proposed implementation is "correct."

This approach in this paper differs from most other work studying formal models for communication by our insistence on being able to ascribe "content" to messages sent in concrete systems of simple agents. To do this, we build upon another theory that shares the same basic philosophy; namely the model used in distributed systems (e.g., [CM86, DM90, HM90]) and artificial intelligence [RK86] for ascribing *knowledge* to agents. We review aspects of this theory in Section 2.

One key idea arising from this work on knowledge in distributed systems is that of *knowledge-based protocols* [CM86, DM90, HZ87]. A knowledge-based protocol is a program or specification that gives preconditions for agents' actions in terms of knowledge. For example,

If $\neg(K\varphi \vee K\neg\varphi)$ **then** *Send-message*(m)

instructs an agent to send a message m if it doesn't know whether some assertion φ is true or not.[1] (Pre-

[1] In this example, $K\varphi$ is interpreted as "the agent knows

sumably, whoever receives m is supposed to respond by returning information about φ.)

We want to extend the theory of knowledge ascription so that we can also ascribe semantic content to the messages sent between agents. One motivation for this concerns knowledge-based protocols. As we have seen, these describe the preconditions for agents' actions in an abstract and intuitive way. But the actions themselves, which typically involve sending messages, are specified quite concretely. This is a curious mix of high-level specification using knowledge and specific implementation details. (Moses [Mos92] also discusses this issue.) The problem seems to be easy to fix, at least conceptually. Instead of describing the agents' actions in terms of which bits are actually sent, we should describe messages in terms of what they tell other agents in the system.[2] The above protocol might be presented more intuitively as:

If $\neg(K\varphi \vee K\neg\varphi)$ **then** $\mathit{Tell}(\neg(K\varphi \vee K\neg\varphi))$.

One goal of this paper is to look at how we might formalize this, by understanding what it means to *tell* other agents something, and how this abstract message content relates to the real messages agents send.

How hard is it to ascribe content to messages? One intuition concerning messages is that they convey information about the world. So it might appear as if any logic for reasoning about the state of a system could be adopted as a "logic of message content." However, we show that things are not always this simple. In Section 2, we introduce a model for message-passing systems, and in Section 3 we look at what "content" might mean in this context. It turns out that, in certain types of system, messages contain information about the *identity* of the sender. We argue that a satisfactory formalism should give semantics to an indexical pronoun "I" for talking about the sender itself. This claim is related to work in [Les91, GH91] which argues for a similar indexical name in logics of knowledge.

However, once the system attains a certain level of complexity even "I" will not be enough: messages can also contain information about how the sender identifies the recipient. We discuss this in more detail later and show that one solution is to add an additional indexical, which we call "You" because it refers to the recipient. These two indexicals are ubiquitous in natural language, and there are various philosophical theories surrounding them; our contribution is to show how they arise out of attempts to understand message content in a very simple and concrete context. We can also use the model to delineate the conditions under which indexicals like "You" play a significant role. It is interesting to note that "You" does not play a role in the knowledge of agents *per se*. The language and semantics for message content that we give is therefore a strict extension of the corresponding epistemic logic: the assertions that messages make are captured in a richer formal language than that used for knowledge.

The logic with the two indexicals "I" and "You" is not the end of the story. As we consider increasingly general systems, other subtleties arise. Each restriction we relax seems to involve a more sophisticated and less natural concept of what a message "means". There does not seem to be a clean answer to the problem of ascribing message content in general. Perhaps the main contribution of this paper is to demonstrate some of the subtleties, and to identify some boundaries (in terms of assumptions we make about the system) beyond which certain difficulties arise.

The final part of this paper, Section 4, discusses two specific applications for a semantic approach to message content. First, we re-examine the idea of knowledge-based protocols, and suggest how to incorporate message content. Second, we discuss how a theory of message content can be used to give a new account of "honesty", which we contrast with the definition used by Fagin and Halpern in [FH88].

2 Preliminaries

2.1 Message-passing systems

We use a relatively simple model for multi-agent message-passing systems, much of which is based on similar definitions in [CM86, DM90, HM90] and elsewhere. We begin by assuming there is a finite set of *agents*, A. These are the individuals we are reasoning about, such as robots or nodes in a distributed system. Although it is slightly restrictive, we assume that the set of agents in a system is fixed.

At any point in time, each agent in A is in a particular internal ("local") *state*. We can think of a state as consisting of values in registers, information about the program being executed, perhaps a clock, and so on. The set S includes all possible agent states.

Agents interact by sending discrete *messages*, from some set M. In many cases the messages can be thought of simply as bit strings, although we will not be using this fact. Indeed, we completely ignore any structure the set M might possess.

How do agents send messages? In our model, we consider agents that can choose among several different communication channels. The key idea here is that of a *port*, which is a particular distinct way to send (or receive) messages. Examples include:

- an "audio" port—i.e., saying something out loud, or hearing a message,
- radio transmission frequencies,

φ." The notation we use later is similar, although more complex.

[2] Moses, in [Mos92], proposes a more radical solution—eliminating explicit communication actions entirely.

- separate lines (wires) connected to nodes in a distributed system,
- a postal or electronic-mail address.

Let P be the set of all ports in the system. We suppose that each agent a has a fixed set of outgoing ports, $Out(a) \subseteq P$ and another (not necessarily disjoint) set of incoming ports $In(a) \subseteq P$. A *channel* is a tuple (a, p_a, b, p_b), where $p_a \in Out(a)$ and $p_b \in In(b)$. We can think of a channel being as actual communication path, which a and b access using p_a and p_b respectively. Notice that one port can access many channels (for broadcast or multicast messages). We say a message is sent (or received) on a channel if it is sent (received) on the corresponding port.

Using this framework, we consider how the system might evolve through time. In this paper time is taken to be discrete, and so is modeled by the natural numbers $\mathbb{N}$. At each time point an agent receives some messages, enters a new local state, and then sends some messages. A *global state*, g, models the state of the entire system at one moment of time. Formally, we take a global state to be a mapping from A to triples $(s, Rec, Sent)$ where $s \in S$ and $Rec, Sent \subseteq (M \times P)$. By $g(a) = (s, Rec, Sent)$ we mean that agent a has just received the messages in Rec (so if $(m, p) \in Rec$ then he received m along port $p \in In(a)$), entered the new local state s, and sent the messages in $Sent$.

A *run* r of a system has two components: a set of channels $Chan(r)$, and a consistent mapping from $\mathbb{N}$ to global states. (We explain the notion of consistency below.) A pair $pt = (r, t)$, where $t \in \mathbb{N}$, is called a *point*; *Points* is the set of all points. In our model, a run represents one possibility for how the system state might change with time. In fact, we formally define a *system* to be a set of runs. This is a reasonable definition, because by specifying all possible behaviors we have captured all observable properties. In practice we usually describe the set of runs implicitly using some compact specification. The most natural description is to list the system's components (agents, states, ports, messages), the possible initial configurations, the protocol the agents follow, and the way the channels work.

Not every run models a legitimate system behavior, so we must restrict attention to runs that are consistent with our common-sense understanding of agents and communication. To begin with, every message received should be preceded by a corresponding send along an appropriate channel. That is, suppose a has received m along port p at the point (r, t). Then there must exist an a' who sent m along some port p' in the same run but at an earlier time. Furthermore, (a', p', a, p) must be in $Chan(r)$. If we also require that the message be sent in the immediately previous time step (i.e, at the point $(r, t-1)$) then we say communication is *round-based*. In round-based communication every message reaches its destination in one step if it arrives at all. To simplify the results in this paper, we always assume round-based communication unless indicated otherwise.

We also want to enforce the idea that each agent is, independently, following some deterministic program.[3] To do this, we assume that there is a state-transition function, $\nu : S \times \mathcal{P}(M \times P) \to S$, which determines the new state an agent enters as a function of its previous state and the messages it has just received.[4] That is, if $\nu(s, Rec) = s'$ and an agent who was in s at the previous time step receives precisely the messages in Rec, this agent should enter state s'. Similarly there must be some fixed communication protocol, which determines the messages an agent sends as a function of its current local state.

There are two other restrictions that we make use of occasionally, but do not assume in general:

- *No forgetting.* A system has no forgetting if, from any agent's local current local state, it is possible to recover both the agent's previous state and the last messages the agent received.
- *Knowing the time.* Agents know the time if any state in S appears in points (r, t) for at most one time t. One way of ensuring this is if each agent has a "clock" as part of its local state.

2.2 Knowledge—Semantics

In the model just described there is no concept of *knowledge* built in. What else is necessary before we can talk about what agents know? The answer to this question, found in such work as [CM86, DM90, HM90], can be divided into several stages. Our review is very brief, and some familiarity with the area is assumed. The few nonstandard details in this section and the next are taken from [GH91, Gro92].

The first stage is purely semantic, and is an instance of the well-known idea of *possible-worlds* models for knowledge. Suppose that the system is actually at point pt, and that agent a is in state s. Let us first suppose that a is the only agent who could ever be in state s (for example, a might have a unique identifier or "name" that is always part of its state). What does a know? Consider any other point pt' in which a is also in state s. So far as a can tell, the system could really be at pt'—a simply cannot distinguish pt from pt' (if it has some way of making the distinction, the relevant extra information should be included in a's state). On the other hand, if pt' is a point in which a would *not* be in s, then presumably a can recoguize

[3] The assumption of determinism is made to simplify the statement of later results. We could present our work in the context of nondeterministic programs, but little more of interest would be added.

[4] We use $\mathcal{P}(X)$ to denote the power set of a set X.

that pt' is impossible. This reasoning leads to a well-known model of knowledge as the *set of points* that an agent considers possible; here this is the set of points in which a is indeed in state s.

Things are more complex if more than one agent could be in state s. Consider a point (r,t) in which another agent a' is in s. Should a consider (r,t) possible? The answer depends on whether a *knows who he is*—if he does not, he should consider (r,t) possible because it is consistent with the fact that "some agent is in s." So in general we must be careful to take into account agents' knowledge about who they are. This issue, which can become quite complex, is discussed in great detail in in [Lew79, Les91, GH91, Gro92]. However, in this paper all we really need is the main conclusion from this work. This is that knowledge is better modeled as a set of (point, agent) *pairs* (i.e., as a member of $\mathcal{P}(Points \times A)$, which we denote KS). More specifically, $ks(s) \in KS$ consists of all (pt', a') such that a' is in state s at pt'.[5] We would say that someone in state s *thinks the system might be at point* pt' (just as before) and, furthermore, *thinks that he might be* a' (in point pt'). Adopting this more complete model of knowledge from the start will make the transition to a logic for reasoning about message content much easier.

Later we make use of the idea that one knowledge state can contain *more* knowledge than another. There are many reasonable ways to formalize this; we use two. The first, and most obvious, is to order KS by set inclusion. If $ks \subset ks'$ then ks certainly represents more knowledge because it rules out more possibilities. However this condition can be too strong. In particular, it is inconsistent with the idea that an agent might know more in some circumstances than it does in others. (It is easy to check that if $s_1 \neq s_2$ are any two states an agent might be in, then $ks(s_1)$ and $ks(s_2)$ will be incomparable under $\subset$.) For example, using $\subset$ we are never able to say that an agent learns more by receiving a message than it would if the message did not arrive (because in the latter case it learns that "the message did not arrive"). Similarly, if an agent discovers that a certain proposition φ is true we cannot say that it then has more knowledge (intuitively, this is because it no longer knows that it doesn't know φ). It seems that $\subset$ misses a useful distinction between knowledge about the "external world" and (often less interesting) knowledge solely concerned with the agent's local state. If knowing that "the message did not arrive" conveys no information about the rest of the system, perhaps it should be ignored when comparing knowledge states. Similarly, knowledge about one's own ignorance is less interesting than new knowledge about other agents. Later, we need to compare knowledge states in a way that ignores "irrelevant" knowledge such as "the message did not arrive". There are several reasonable definitions that allow this. We use one that is based on $\subset$, but which ignores differences in possible (point, agent) pairs that are entirely local to the agent's current state. Formally, we have:

Definition 2.1: Two pairs $((r_1,t),a)$ and $((r_2,t),a)$ are *essentially equivalent*, written $((r_1,t),a) \approx ((r_2,t),a)$, if r_1 and r_2 are identical in all respects for times $\leq t$, except that the state of agent a at time t can differ in the two runs.[6] We say that a knowledge state ks_1 is *essentially as strong as* ks_2, written $ks_1 \preceq ks_2$, if every element of ks_1 is essentially equivalent to some element of ks_2. Finally, $ks_1 \approx ks_2$ if both $ks_1 \preceq ks_2$ and $ks_2 \preceq ks_1$. ∎

2.3 Knowledge—A logic

The next stage towards a theory of knowledge is to construct a formal language. To do this, we first need a way of referring to properties the system might possess (the propositions), and we need to decide how the agents refer to each other (the names).

Definition 2.2: An *interpreted system*, $\mathcal{IS}$, is a system (as defined in Section 2.1) together with the following.

- A set Pr of proposition symbols. These will be used to talk about interesting properties of the system.
- A truth assignment function π. The function $\pi : Points \times A \times Pr \to \{true, false\}$ interprets the propositions, by telling us at which (point, agent) pairs they are true.
- A set N of *names*. These are the symbols used to refer to an agent or a set of agents.
- A name interpretation function $\tau : Runs \times A \times N \to \mathcal{P}(A)$. By $\tau(r,a,\mathbf{n}) = B$ we mean that, in run r, agent a uses the name $\mathbf{n}$ to refer to the set of agents B.

Because π and τ depend on an agent as well as a point or run, this definition allows *indexical* propositions and names (i.e., propositions and names whose truth value or denotation depends on the identity of the agent we are reasoning about). The proposition "I have red hair" and the name "The person sitting beside me" are indexical. *Objective* propositions and names, whose denotation is independent of the agent, are simply a special case of the definitions given. For instance, the proposition "The sun is shining" and the

[5] Strictly speaking, $ks(s)$ depends on the system we are investigating. We suppress this dependence in this and later notation.

[6] The possibility that the agent's internal state can differ at t might have repercussions for the rest of the system later on, if the difference affects the messages that the agent decides to send. This is why we do not require agreement for times after t.

name "The president of the United States" are objective in this sense.

The names and propositions used are chosen by us, as external observers, to help us reason about the system. In principle we can choose any collection of names and propositions whatsoever, although some choices will turn out to be less effective than others. However, there are some special names that are so generally useful that we assume that they are always present. To understand what these names are, notice that agents often need to reason about "the set of all agents on the other end of channels connected to a particular port." So we need a way of referring to agents' associated ports within the logic. To do this, we assume a fixed class of *port names* $PN = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \ldots\}$. In any run r, an agent a associates each of his ports $p \in In(a) \cup Out(a)$ with some port name $\sigma(r, a, p) \in PN$. To simplify things further, we assume that (1) agents associate different names with distinct ports, and (2) every agent "knows" the names it assigns.[7] In our logic we use port names just as any other name, to denote a set of agents. Roughly speaking, relative to agent a the name $\mathbf{p}_i$ denotes all agents on the other end of channels which a accesses using the port associated with $\mathbf{p}_i$. We give the formal definition for the case of a port $p \in Out(a)$. If $\sigma(r, a, p) = \mathbf{p}_i$, we have

$$\tau(r, a, \mathbf{p}_i) = \{a' : \exists p' \textit{ with } (a, p, a', p') \in Chan(r)\}.$$

There are two things to note about port names and the mapping σ. First, these names are indexical—a uses $\mathbf{p}_i$ for the set of agents attached to one of *its* channels; for other agents, the same name refers to one of *their* ports. Even for a fixed agent a, the definition allows the correspondence between a's port names and the ports themselves to vary from run to run. The reason is that other agents might not know what names a uses, and so will consider several different mappings between a's ports and names to be possible.[8]

Another special name we use is I. This has a fixed denotation: $\tau(r, a, \mathbf{I}) = a$ always. Thus, I gives every agent a convenient way of referring to "himself."

It is also useful to include some compound propositions and names. First, if $\mathbf{n}_1$ and $\mathbf{n}_2$ are names we will assume the *composition* $\mathbf{n}_1 \circ \mathbf{n}_2$ is also. Relative to (pt, a), it denotes all agents that are named $\mathbf{n}_1$ by some agent whom a names $\mathbf{n}_2$. Second, given two names, the expression "$\mathbf{n}_1 \subseteq \mathbf{n}_2$" is a legitimate proposition. It is true if the denotation of $\mathbf{n}_1$ is contained in that of $\mathbf{n}_2$.

Finally, we use the vocabulary of names and propositions as the heart of a formal language for reasoning about agents' knowledge. We have in mind a fairly standard modal language:

- All propositions in Pr are formulas.
- If $\mathbf{n}$ is a name, both $K_{\mathbf{n}}^{\forall}\varphi$, read as "every agent named $\mathbf{n}$ knows φ," and $K_{\mathbf{n}}^{\exists}\varphi$, "some agent named $\mathbf{n}$ knows φ" are formulas. If $\mathbf{p}$ always denotes just one agent, we drop the superscript.
- $\bigcirc\varphi$, read "at the next time step φ", is a formula, as is $\bigcirc^{-}\varphi$, "at the previous step φ." Other temporal modalities can be added if desired.
- Boolean combinations of formula are allowed.

The clauses relating the language to our semantics (interpreted systems) are also fairly standard. Some of the relevant definitions include:

- (Propositions in Pr.) $\mathcal{IS}, (r, t), a \models \mathbf{q}$ if $\pi((r, t), a, \mathbf{q}) = true$.
- (Name comparisons.) $\mathcal{IS}, (r, t), a \models \mathbf{n}_1 \subseteq \mathbf{n}_2$ if $\tau(r, a, \mathbf{n}_1) \subseteq \tau(r, a, \mathbf{n}_2)$.
- (Everybody knows.) $\mathcal{IS}, (r, t), a \models K_{\mathbf{n}}^{\forall}\varphi$, if, for all $a' \in \tau(r, a, \mathbf{n})$, we have $\mathcal{IS}, ks(s) \models \varphi$, where s is the local state of a' in (r, t). This notation means that φ must be true at every (point, agent) pair in $ks(s)$.
- (Temporal modalities.) $\mathcal{IS}, (r, t), a \models \bigcirc\varphi$ if $\mathcal{IS}, (r, t+1), a \models \varphi$.

3 Message content

A theory of message content should tell us how to ascribe a semantic structure of some sort, the *content,* to each message in a system, and should provide a logic for reasoning about the chosen semantics. There are very general two criteria we can use when evaluating proposed theories. First, the content ascribed to a message m should reflect the role m actually plays in the system. Second, the theory should be simple and intuitive. Unfortunately, as we discuss shortly, these conditions are in partial conflict.

It is often said that messages are important because they act as *state transformers.* To understand this, recall our assumption that agents' actions are determined by some fixed state-transition function. Shifting viewpoint slightly, we can regard this function instead as a *family* of state-transition functions, each indexed by possible messages that an agent can receive (and, of course, the ports they are received on). The function $\nu_{(m,p)}$ describes how agents change state upon receiving message m from port p. At first glance, $\nu_{(m,p)}$

[7] Formally, let $((r, t), a)$ and $((r', t'), a')$ both be in $ks(s)$ for some state $s \in S$. Then the mapping σ between ports and port names must be the same for a in r as for a' in r'.

[8] It is somewhat imprecise to speak of "the names an agent uses" since names are ascribed by us, not the agents. However, the agent's programming will cause it to use its ports in certain ways, and it is useful to choose names that reflect this. In this sense, some names are indeed implicit in the system, in the ways ports are used.

looks like an excellent candidate for the "content" of message m, when received along p. It captures every property of the message's reception that is relevant to the evolution of system we are studying. So the state-transformer semantics for message content is as powerful as we could hope for. A suitable logic for this semantics would probably be related to *dynamic logic* [Har79].

Nevertheless this is not the approach we take in this paper, because we would prefer a simpler and more intuitive theory. Consider the example in the introduction, concerning an agent a who sends a message m precisely when he is uncertain about the truth of an assertion φ. Is the content of m "I don't know whether φ"? Under the state-transformer semantics the answer might be no, because the content will reflect the recipients' reactions as well. For instance, if a recipient sometimes discards the message without reading it, the ascribed content must reflect this. This is somewhat unintuitive. The notion of message content we use in informal reasoning seems to involve relatively straightforward assertions such as "I don't know whether φ", and not functions from states to states that take into account all possible ways the recipient might process the message.

How can we get a more natural theory? Our answer is to settle for a weaker connection between a message's content and the effect it has on its recipients. Instead of trying to capture precisely how a recipient reacts to the message, we look for theories of message content that tell us how much knowledge the recipient could *possibly* gain (i.e., if it were to extract the maximum possible information from the message). When we judge a theory against this criterion—that the ascribed content only needs to capture, somehow, the knowledge that an agent could gain—there are at least two interesting outcomes.

- *Inadequacy.* A theory is inadequate if there is some knowledge transfer that the theory cannot explain. One way of demonstrating inadequacy is to find two messages that are ascribed the same content, but which are capable of carrying different information to an agent who might receive them.
- *Completeness.* Conversely, we would like to be able to say when a message's content captures *all* interesting information that might be gained. One way of making this precise is to show that the content of a message is enough to compute some (preferably tight) bound on the knowledge resulting from message reception.

It turns out that the most straightforward and intuitive theories of message content are inadequate unless we consider very restricted types of system. This tradeoff between generality and simplicity is the main theme of much of this section.

3.1 Singly-connected systems—semantics

We begin in a fairly simple setting, because it simplifies the presentation of some key ideas. Throughout this subsection, we consider systems that we call *singly-connected*. A system is singly-connected if, in any one run, there is at most one channel connecting any two agents. Simple systems are often singly-connected—it can be inefficient to provide more than one channel between two points. It turns out that, by making this assumption, we can discuss message content largely in terms of the agents themselves (whereas, as we see in Section 3.3, a general treatment requires more explicit reasoning about channels and ports).

For the moment, we will make one other assumption—that every channel is *unreliable* in the following sense:

Definition 3.1: *Unreliability.* A system is unreliable if the following is true, for all messages m. Suppose m was sent over the channel (a, p_a, b, p_b) at point (r, t). If a receives (resp., does not receive) m at $(r, t+1)$, then there is another run r' which is identical to r in all respects up to time $t+1$, except that a did not receive (resp., did receive) m at $(r', t+1)$. ∎

Unreliability does not simply mean that messages sometimes fail to arrive. For now, we do not want to consider systems in which an intelligent adversary decides whether to deliver messages or not. The problem with such systems is that an agent might gain useful information just from the fact that m was delivered, which might bear no interesting relation to the knowledge and goals of the messages's sender. The definition just given avoids this, by requiring that messages are delivered (or not) independently of other properties of the system.

For later reference, we also define what it means for a communication in a system to be reliable.

Definition 3.2: *Reliability.* A system is reliable if the following is true, for all messages m. If m is sent over a channel (a, p_a, b, p_b) at point (r, t), then b must receive m at $(r, t+1)$ along p_b. ∎

Reliable channels are a bit harder to deal with than unreliable channels. The problem with reliable channels in synchronous systems an agent gains useful information when he does *not* receive a message. We discuss this issue later.

The approach we take in this section is to look at three proposals for a "semantics for message content" for singly-connected channels with unreliable communication. Each proposal is more powerful, although more complex, than the previous one. In Section 3.2 we present a logic for reasoning about the most powerful of the three semantic models.

Our first intuition about a message m is that it tells its recipient something about the state of the system when

m was sent. This leads to the following definition:

Definition 3.3: *Objective semantics.* Let $\mathcal{IS}$ be an interpreted system, and m a message in $\mathcal{IS}$. The *objective content* of m, $os(m)$, is the set of all points in which m was sent. ∎

For instance, if my alarm clock rings each day at six o'clock, the objective content of the alarm is "It is six o'clock." The objective semantics is very natural but is also quite weak. Problems arise whenever there is doubt about who sent a message, who might receive a message, or what channels connect the sender and recipient. Consider the following example:

Example 3.4: Suppose that, at time 0, exactly one of agents a or b knows a fact φ. The agent who knows φ sends message m and the other agent sends m'. Suppose c receives m along port p. It can surely learn that the agent who is connected by p—either a or b, although c doesn't necessarily know which one—knows φ. Of course, c would have learned something else had m' arrived instead. The objective semantics for message content cannot explain the difference between the two cases, because $os(m) = os(m')$. (Both contents are just the set of all time 0 points, because both messages are always sent by someone.) ∎

The main problem here is that the sender's identity is not fixed. We do not simply want to reason about the set of points where m was sent—we also want to reason about who the sender was. This motivates the following definition:

Definition 3.5: *KS-semantics.* Let $\mathcal{IS}$ be an interpreted system, and m a message in $\mathcal{IS}$. The *KS-content* of m, $ks(m)$, is the set of pairs (pt, a), such that agent a sends m at point pt. ∎

Notice that message contents using KS-semantics are the same structures we use for modeling agent's knowledge states (hence the notation). This is intuitive: we can interpret it as saying that messages convey some of the sender's knowledge, which includes knowledge about his identity as well as "objective" knowledge. Furthermore, the KS-semantics is reasonably powerful. It cope with the previous example; the content of message m is basically "I know φ". We say that a message m is *addressed* if it is always received by the same agent (i.e., in every run). For addressed messages, the KS-semantics is *complete* in a certain sense, which is formalized in the following theorem. The main idea of this theorem is that if an agent receives m then we can use m's content to calculated a bound on the agent's resulting knowledge state. This bound is also a function of the agent, its previous state, and the port it received the message on. But the main point is that the only property of the message itself that we need is the content—all other properties have been successfully abstracted away.

Theorem 3.6: *Let $\mathcal{IS}$ be any singly-connected interpreted system with unreliable message delivery, in which all messages are addressed. There exists a function $\beta : A \times KS \times KS \times P \to KS$ with the following property.*

Suppose a, in state s_{before}, receives the single message m with content $ks(m)$, along port p. The subsequent state of a, s_{after}, satisfies

$$\beta(a, ks(s_{\text{before}}), ks(m), p) \preceq ks(s_{\text{after}}).$$

We can replace $\preceq$ by $\approx$ in any system with no forgetting, in which agents know the time.[9]

Proof: Here, $\beta(a, ks(s_{\text{before}}), ks(m), p)$ consists of all pairs $((r,t), a)$ such that there exists a', with (1) $((r, t-1), a) \in ks(s_{\text{before}})$, (2) $((r, t-1), a') \in ks(m)$, and (3) a' is connected to a via port p in r.

Consider any $((r,t), a) \in \beta(a, ks(s_{\text{before}}), ks(m), p)$. We know that a was in state s_{before} at $(r, t-1)$, by definition of $ks(s_{\text{before}})$. Also, by our definition of message content, m was sent by a' at $(r, t-1)$. While it is possible that this message was never delivered, the assumption of unreliable communication ensures that there is another point, (r', t), which is identical to (r, t) in all respects before time t except that m was in fact delivered. Because m is addressed, the message is delivered to a. Notice that $((r', t), a) \approx ((r, t), a)$. By the third clause in the definition of β and the assumption of single-connectedness, the message is delivered along port p. By determinism, a must enter state s_{after} in run (r', t), and so $((r', t), a) \in ks(s_{\text{after}})$ as required.

Now, suppose that the system has no forgetting and agents know the time. Consider a pair $((r,t), a'')$ in $ks(s_{\text{after}})$ (that is, the state of a'' at (r,t) is s_{after}). By the definition of no forgetting, we know that a'''s previous state was s_{before}, and the last message it received was m (along port p). Because m is addressed, we must have $a'' = a$. Because a knows the time, it must have received this message in the current time round. Therefore, $((r,t), a'') \in \beta(a, ks(s_{\text{before}}), ks(m), p)$. The result follows. ∎

We remark that this theorem can be easily restated to take account of points in which a receives more than one message. On the other hand, the assumption that messages are addressed is important. Consider the following two examples:

[9] In general, we do need to use $\preceq$ (and not $\subseteq$) for comparing knowledge states in this result. In addition to the knowledge about the system state that m conveys, the recipient also learns the bare fact that "I just received a message." By itself, this is just not interesting—because communication is unreliable, it does not tell the agent anything additional about the rest of the system. Our definition of $\preceq$ ignores such purely internal knowledge.

Example 3.7: Suppose that in every run, at time 0, agent a sends two messages: m is sent along a fixed port p_a, and m' is sent along whichever port (which may or may not be p_a) is connected to another agent b. Notice that m is not addressed, because a's port p_a might have been connected to someone else. If agent b receives m along p_b he learns that (a, p_a, b, p_b) is a channel in the system. However b cannot learn this if he receives m' instead. Nevertheless, KS-semantics gives both messages the same content: the set of all pairs $((r, 0), a)$, for every run r in the system. Therefore, the KS-semantics is inadequate in this case. ∎

Example 3.8: Suppose agent a sends two messages: the first, m, is via a port connected to both b and c, whereas the second message, m', can only be received by d. Suppose that, before these messages were sent, all of b, c and d were in the same state. Note that these three agents are not even certain of their own identity—for instance, b thinks it might be d, because it doesn't have any information to the contrary. Later, if b receives m, it learns something: that it isn't d (it will still think that it might be c). It would not learn this if it had received m'. Nevertheless, both m and m' are ascribed the same content by KS-semantics. ∎

In the first of these examples the recipient of m learned something about how it is connected to other agents, and in the second it learned something about its own identity. A powerful semantics for message content needs to reason about who the recipient might be, as well as who the sender is. To do this, we modify KS-semantics by making the recipient explicit as well:

Definition 3.9: *Double-indexical semantics.* Let $\mathcal{IS}$ be an interpreted system, and m a message in $\mathcal{IS}$. The *double-indexical content* of m, written $c(m)$, is the set of all triples (pt, a, a') such that agent a sends m at point pt, along a channel connected to a'. In the following, we use C to denote $\mathcal{P}(Points \times A \times A)$. ∎

This semantics is very powerful, coping easily with the two previous examples. We get an analogue to Theorem 3.6 in which messages do not need to be addressed.

Theorem 3.10: *Let $\mathcal{IS}$ be any singly-connected interpreted system with unreliable message delivery. There exists a function $\beta' : KS \times C \times P \to KS$ with the following property.*

Suppose a, in state s_{before}, receives the single message m with content $ks(m)$, along port p. The subsequent state of a, s_{after}, satisfies

$$\beta'(ks(s_{\text{before}}), c(m), p) \preceq ks(s_{\text{after}}).$$

We can replace $\preceq$ by $\approx$ in any system with no forgetting, in which agents know the time.

Proof: We take $\beta'(ks(s_{\text{before}}), c(m), p)$ to consist of all pairs $((r, t), a')$ such that (1) $((r, t-1), a') \in ks(s_{\text{before}})$, and there exists another agent a'' such that (2) $((r, t-1), a'', a') \in c(m)$ and (3) a'' is connected to a' by port p in r. The rest of the proof is similar to that given for Theorem 3.6, and is omitted. ∎

The double indexical semantics are also appropriate in systems with reliable channels. In such systems, agents gain useful knowledge from the *non-arrival* of messages. (If a would send me a message if he knew φ, and I don't get a message, then I learn that a did not know φ.) We can deal with this quite easily—all we need to do is regard "silence" simply as another message! The content of "silence" is just all triples $((r, t), a, a')$ for points (r, t) in which a does not send anything to a'. Of course, "silence" is usually not an addressed message, and so it is not entirely surprising that this strategy does little good when we use KS-semantics. But with double-indexical semantics, we can prove an analogous theorem to the above. We omit details in the interest of brevity.[10]

3.2 A logic for singly-connected systems

The semantic analysis just given is only half the story—we also want a formal logic for reasoning about messages and agents. In this section, we give a brief account of how this might be done using double-indexical semantics. Of course, many different logics can be constructed based on a single semantic model; there is a tradeoff is between expressive power and simplicity. As our main goal is to understand some of the basic principles, the specific logic we present emphasizes simplicity. (See also [FV86, Bie90] for some related discussion of properties of knowledge in systems of communicating agents.)

Our new logic is an extension of the epistemic logic discussed earlier. In addition to being able to reason about knowledge, we want to talk about sending and receiving messages, and the content of these messages. We do this by adding new modal constructs:

- $R^{\exists}_{\mathbf{n},\mathbf{p}}\varphi$, read as "some agent with name $\mathbf{n}$ has just received a message with content φ, along the port it calls $\mathbf{p}$." We discuss the form of φ shortly.
- $S^{\exists}_{\mathbf{n},\mathbf{p}}\varphi$, read as "some agent with name $\mathbf{n}$ has just sent a message with content φ, along the port it calls $\mathbf{p}$."
- Analogously, $R^{\forall}_{\mathbf{n},\mathbf{p}}\varphi$ and $S^{\forall}_{\mathbf{n},\mathbf{p}}\varphi$ talk about *all* agents with a given name.

What is the syntax of formulas φ appearing after one of these operators? These formulas should talk about

[10] There are two main differences in the reliable channel theorem. First, the bound on knowledge gained is a function of *all* the messages sent, including silences. Second, we can use the simpler $\subseteq$ definition for comparing two knowledge states when stating the bound.

a message's content, which is a set of (point, agent, agent) *triples*. Recall that, in the logic for knowledge, formulas are interpreted over (point, agent) pairs. In that logic we use indexical names and propositions that are interpreted relative to the agent (the "knower"). So it seems reasonable that a logic for message content should have more indexical symbols; some will be interpreted relative to the sender and others relative to the recipient.[11]

We formalize this as follows. Suppose we have fixed a vocabulary Pr and N for reasoning about knowledge, together with the interpretation functions π and τ. Our vocabulary for message content includes all these symbols, and they are interpreted relative to the sender (the first agent in a triple). For instance, relative to (r, a, a'), the name $\mathbf{n}$ denotes $\tau(r, a, \mathbf{n})$. Notice that the name $\mathbf{I}$ refers directly to the sender (a). However, we also add new names to be interpreted relative to the recipient. For each name $\mathbf{n}$ in N (resp., proposition $\mathbf{q} \in Pr$) we allow a new symbol, $\mathbf{n}^Y$ (resp, $\mathbf{q}^Y$). The denotation of $\mathbf{n}^Y$ at (r, a, a') is just $\tau(r, a', \mathbf{n})$. The name $\mathbf{I}^Y$, which for convenience we write as **You**, refers directly to the recipient of a message.

The new language is constructed in a straightforward way, extending the underlying epistemic logic. Essentially, we begin with the basic propositions and name comparisons (using $\subseteq$), and close under the boolean connectives and all of the modal operators (epistemic, message, and temporal). There is one extra condition. The new symbols (those with superscript Y) have been given a meaning only when we are reasoning about the content of a message. So these symbols should only appear in contexts where they clearly make sense—immediately after one of the four new message modal operators.[12] It follows that the class of formulas that can appear after a message operator strictly contains the class of formulas that appear after knowledge operators. This just reflects the fact that knowledge and content are different semantic objects.

It should be clear how the new language is interpreted over our semantics, because we have already discussed how the names and basic propositions are interpreted relative to (point, agent, agent) triples. The truth conditions for the new modal operators are also straightforward. For instance:

- $\mathcal{IS}, (r, t), a, a' \models R^{\exists}_{\mathbf{n},\mathbf{p}}\varphi$ if there is some a'' in the denotation of $\mathbf{n}$ at $((r, t), a, a')$ who has just received a message m, along the port he associates with the name $\mathbf{p}$, and $\mathcal{IS}, c(m) \models \varphi$. (That is, φ is true at each one of the triples in the content $c(m)$).

We do not yet have a complete axiomatization or decision procedure for this logic. Axiom systems for temporal and epistemic reasoning are well known. However, there will be additional rules for dealing with the message modalities. For instance, the axiom

$$S^{\exists}_{\mathbf{I},\mathbf{p}}\varphi \Rightarrow K_I S^{\exists}_{\mathbf{I},\mathbf{p}}\varphi$$

(if you send a message, you know you are sending it) turns out to be valid. Another fairly straightforward axiom that is valid only in systems with reliable communication is

$$S^{\exists}_{\mathbf{I},\mathbf{p}}\varphi \Rightarrow \bigcirc R^{\forall}_{\mathbf{p},\mathbf{Someport}}\varphi,$$

where **Someport** is an special indexical associated with all possible incoming channels. In words: If I send a message over port $\mathbf{p}$, the agents on this port will receive it (over one of their ports—but I don't necessarily know which) one time step later.

The most interesting new axioms are the "substitution" principles. Consider an agent a sending a message m whose content, φ, refers to the indexical **You**. We might expect a to actually *know* φ, but this cannot be the case because φ refers indexically to the recipient. (In the logic we have defined, φ does not have any semantic meaning at all unless we are reasoning about the content of a message.) However, it is true that a knows something that is closely related to φ. Suppose the message is sent along the port a thinks of as $\mathbf{p}$. Then a doesn't think of the message's recipient simply as **You**, but instead uses the name $\mathbf{p}$. This suggests that the formula φ', formed by replacing all occurrences of **You** by $\mathbf{p}$, is indeed known by the sender. Somewhat more generally, let $\mu^{\mathbf{p}}_1(\varphi)$ be formed from φ by replacing all indexicals $\mathbf{n}^Y$, that are not within the scope of any modality in φ, by $\mathbf{n} \circ \mathbf{p}$. Then, so long as the name $\mathbf{p}$ refers to a single agent and if φ has no indexical propositions, the formula

$$S^{\exists}_{\mathbf{I},\mathbf{p}}\varphi \Rightarrow K_{\mathbf{I}}(\mu^{\mathbf{p}}_1(\varphi))$$

is indeed valid.[13]

There is a corresponding rule for message reception, although it is valid only in systems with no forgetting. If an agent receives a message with content φ, we need to replace all the indexical references to the sender by the the actual port the message was received on. We define $\mu^{\mathbf{p}}_2(\varphi)$ to be like φ, except that all indexicals $\mathbf{n}^Y$ are replaced by simply $\mathbf{n}$, and all the other names

[11] It is also possible to have indexicals that are interpreted relative to both agents simultaneously. For example, we might have a name **We** that refers to the set of both the sender and recipient. To keep things simple we do not discuss this possibility any further in this paper.

[12] By "immediately after", we mean that there can be no intervening knowledge modalities. It turns out that temporal modal operators do not cause any problem.

[13] The generalization to names denoting many agents and to indexical propositions is not difficult, although the necessary transformation of φ is much more cumbersome.

$\mathbf{n}'$ are replaced by $\mathbf{n}' \circ \mathbf{p}$. Then, in systems with no forgetting,

$$R^{\exists}_{\mathbf{I},\mathbf{p}}\varphi \Rightarrow K_{\mathbf{I}}\bigcirc^{-}(\mu^{\mathbf{P}}_{2}(\varphi))$$

is valid. (Again, this simple form of the axiom needs modification if $\mathbf{p}$ can denote many agents or if φ uses indexical propositions symbols.)

3.3 More general systems

So far we have only considered singly-connected systems. All our definitions make perfect sense even without this assumption, and the logic is still useful. However Theorem 3.10 will fail. Consider this example:

Example 3.11: Suppose there are two channels, (a, p_a, b, p_b) and (a, p_a', b, p_b'), between agents a and b. If a knows a certain proposition φ, it sends m along p_a and m' along p_b; if a doesn't know φ, the ports and messages are reversed. So if b receives m along p_b he learns that a knows φ; obviously, he will not learn this if he receives m' instead. However the double-indexical contents of m and m' are the same: namely, the triples $((r,0),a,b)$ for all runs r. ∎

To regain a completeness result analogous to Theorem 3.10, we need a notion of message content that deals more directly with the ports that agents use. It turns out that sets of triples, (pt, a, p), such that agent a sends the message along port p in point pt, give appropriate semantics.[14] It is not hard to see how this proposal can cope with the example. Because this semantics does not deal with the recipient directly, it is somewhat less intuitive than double indexical-semantics. Nevertheless, it would be straightforward to develop a suitable logic.

Even this proposal is far from being an "ultimate" semantics for message content. So far, the classes of systems we have been considering have been fairly limited—we have made several explicit assumptions and others are inherent to the basic model for systems that we are using. When we drop these assumptions, new subtleties arise. We close this section by briefly mentioning two of the more interesting issues.

- *More complex channel behavior.* In many systems, the channels are neither reliable or unreliable in the sense we have been using these terms. For example, consider a channel which is subject to failures, although once it fails no messages ever get through again. Or suppose we generalized our model so that messages could be delayed for more than one time step (which by itself is not such a difficult extension). We can easily imagine channels in which the delay depends on the number other messages being sent at the time.

  The problem in such cases is that the mere fact that a message arrives can convey interesting information, because we learn about properties of the channel. In the first case above, we learn that every message sent earlier along the same channel should have arrived also. In the second example, receiving a message very soon after it was sent tells the recipient that very few (other) messages are currently being sent, and this might say a lot about the overall system state.

  A semantics for content that treats information gain due to channel behavior is likely to be even more complex than those we have discussed. It is also likely to be fairly system-dependent, because there are many different behaviors we might encounter.

- *Cryptography.* Often agents send *encrypted* messages to enhance security. Suppose an agent receives an encrypted message m, but does not possess the correct decryption key. We would like to say that the agent learns next to nothing from m, because without the key it looks just like a random bit string. What definition of "content" would match this intuition? Perhaps we would like to say such things as "Only if you know the right key will you learn...".

  Unfortunately, the framework we have presented does not deal with encrypted messages very well. The problem is with the model of agents' knowledge that we have been using. As is well known, this logic does not taken into account the fact that agents have limited computational ability. So an agent does not need a key to "understand" an encrypted message—the plaintext is usually implicit in the code. (After all, in principle one could try all reasonable messages and all reasonable keys, and eventually find the, often unique, pair that produces the encrypted message.)

  We need to build a semantics for message content on top of a theory of knowledge in cryptographic settings (see, for instance, [MW85, Bie90]). This is work in progress.

4 Applications

Is a theory of message content useful? Here we discuss two possible applications.

4.1 Abstract protocols

We have already mentioned the concept of *knowledge-based* protocols, where the preconditions for agents' actions (in particular, sending messages) are specified using conditions on the agents' knowledge. We can use an understanding of message content to specify the messages on a similarly abstract level. Not only will such a specification be more intuitive, but it may

[14] Note that this is more general than double-indexical-semantics, because from pt, a, and p we can work out who the possible recipients are.

also allow us to reason about more properties of the system within a formal logic. We illustrate how this extended concept of knowledge-based protocol might be used with an extremely simple example.

Consider agents arranged in a ring—each has two outgoing ports, one connecting to each of its neighbors. Initially, each agent chooses one port as its "left" port, and labels the other as its "right" port. The choices are made independently and so are not necessarily consistent with each other (a's left neighbor might consider a to be on *his* left, for instance).

Sometimes, it is important that the labeling be consistent. As a first step towards attaining consistency, consider how we would design a simple protocol to check for *local* consistency. (We say that the labeling is locally consistent for a if the right neighbor of a's left neighbor, and the left neighbor of a's right neighbor, are both just a itself.) To check local consistency the agents need to communicate with each other. It seems obvious what the necessary messages should say: every agent needs to tell his left (right) neighbor that he *is* his left (right) neighbor. A message saying "You are on my left" should be enough. We can formalize this in a knowledge based-protocol that uses high-level message content, as follows:

$$\textbf{If } K_{\mathbf{I}}\mathit{start} \textbf{ then : } \quad \mathit{Tell}(\textbf{left}, \textbf{You} \subseteq \textbf{left})$$
$$\mathit{Tell}(\textbf{right}, \textbf{You} \subseteq \textbf{right})$$

Here, *start* is true when only when time is right to do the consistency check. The precondition K_I*start* ensures that the agents send the messages only when they are supposed to. *Tell*(**left**, **You** $\subseteq$ **left**) instructs the agent to send a message, along the port it associates with the name **left**, whose content is (at least) **You** $\subseteq$ **left**. In words, "*You* are on *my* left port."

The formula describing local consistency, relative to any agent, is $\mathbf{I} \subseteq \textbf{left} \circ \textbf{right} \wedge \mathbf{I} \subseteq \textbf{right} \circ \textbf{left}$. For brevity, we call this formula *consistency*. Consider a system in which every agent's protocol satisfies the above knowledge-based specification. If communication is reliable then such formulas as

$$K_{\textbf{left}}\mathit{start} \Rightarrow$$
$$\bigcirc(R^{\vee}_{\mathbf{I},\textbf{left}}(\textbf{You} \subseteq \textbf{left}) \vee R^{\vee}_{\mathbf{I},\textbf{left}}(\textbf{You} \subseteq \textbf{right}))$$

are valid throughout the system. (In words: If the agent on my left knows *start*, then at the next step I will receive a message from my left, telling me whether I am on the sender's right or left.) Using such assumptions (as well as other properties of the system, such as that every port is connected to at most one channel, and the assumption of no forgetting) we can conclude *within the logic* that

$$(K^{\vee}_{\textbf{All}}\mathit{start}) \wedge \mathit{consistency} \Rightarrow \bigcirc K^{\vee}_{\textbf{All}}\mathit{consistency}.$$

(**All** is a name that denotes all agents in the system.) That is, the protocol fragment does what we want—if the system is locally consistent, everyone will learn this. If an interpreted system is an *implementation* of this abstract protocol, in the sense that the ascribed knowledge and content agrees with the protocol, it will also satisfy this conclusion. It is easy to verify that a protocol in which the message sent along the left and right channels are distinct, although the same for every agent, is in fact an implementation of the abstract protocol.

4.2 Honesty

Once we have a logic for message content, we can give an account of *honesty*. Suppose messages are in fact formulas in the logic, or can be interpreted as such. Then not only does the message have an ascribed semantic content; it also makes an explicit claim about what this content is. Our suggestion is to simply to say that a message is honest if the ascribed content does, in fact, entail the claim that is being made.

A subtly different definition sometime used is that a message is honest just if the sender knows that the claim made by the message is true (see [FH88]). The main difference between this definition and ours arises because, as we have seen, message content is sometimes more than just an assertion about some agent's knowledge. The definition of honesty based on knowledge cannot cope properly with indexical uses of names like **You**. We have suggested that quite complex notions of content will be needed at times; our definition of honesty generalizes easily whereas the second definition does not.

Suppose that we restrict attention to messages that can be thought of as being about knowledge (for instance, by using KS-semantics). The two definitions are still not equivalent. Consider a run r in which an agent a sends a message m, and knows that the claim being made is true. But suppose that, in fact, a would have sent m no matter what it knew. Is m honest? By our definition it is not, because the content of m reflects all the situations in which m might have been sent. (Since m is always sent, its content is likely to be very uninformative.) According to the other definition, m was sent honestly. Therefore there is a difference between saying that a message is honest, and saying that a particular instance of sending that message was honest. Presumably, both viewpoints are useful in different circumstances; the definition based on content is best suited to the former.

5 Conclusion

We have discussed how one might ascribe "content" to messages in simple systems of interacting agents.

One conclusion is that the issue is unexpectedly complex, and how good a proposed notion of content is

depends on the precise nature of the systems being considered. By working in a formal model of message-passing systems, we can identify the conditions under which any particular theory is appropriate.

Message content does not always concern objective facts about the world. Messages also convey information about the identity of the sender and recipient. We have given a simple theory that deals with this issue satisfactorily in many cases. Our logic uses indexical terms, such as I and You, which have a strong intuitive correspondence to their natural language counterparts. Interestingly, for us these terms and their semantics arise naturally out of of the models we investigate. This contrasts with other studies of indexicals, such as [Kap77], which are typically based on linguistic or philosophical theories of human communication.

Acknowledgments

The author is grateful to Joseph Halpern and Daphne Koller for many helpful discussions, and for comments on early drafts of this paper. The author has been supported by an IBM graduate fellowship, and in part by the Air Force Office of Scientific Research (AFSC), under Contract F49620-91-C-0080. The United States Government is authorized to reproduce and distribute reprints for governmental purposes.

References

[Bie90] P. Bieber. A logic of communication in hostile environments. In *Proc. The Computer Security Foundations Workshop III*, pages 14–22. IEEE Computer Society Press, 1990.

[CM86] K. M. Chandy and J. Misra. How processes learn. *Distributed Computing*, 1(1):40–52, 1986.

[CP79] P. Cohen and C. R. Perrault. Elements of a plan based theory of speech acts. *Cognitive Science*, 3:177–212, 1979.

[DM90] C. Dwork and Y. Moses. Knowledge and common knowledge in a Byzantine environment: crash failures. *Information and Computation*, 88(2):156–186, 1990.

[FH88] R. Fagin and J. Y. Halpern. I'm OK if you're OK: On the notion of trusting communication. *Journal of Philosophical Logic*, 17(4):329–354, 1988. Reprinted in *Philosophical Logic and Artificial Intelligence* (ed. R. H. Thomason), Kluwer, 1989, pp. 9–34.

[FV86] R. Fagin and M. Y. Vardi. Knowledge and implicit knowledge in a distributed environment: preliminary report. In J. Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge: Proc. 1986 Conference*, pages 187–206. Morgan Kaufmann, 1986.

[GH91] A. J. Grove and J. Y. Halpern. Naming and identity in a multi-agent epistemic logic. In J. A. Allen, R. Fike, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proc. Second International Conference*, pages 301–312. Morgan Kaufmann, 1991.

[Gro92] A. J. Grove. *Topics in multi-agent logics.* PhD thesis, Stanford University, to appear, 1992.

[Har79] D. Harel. *First-Order Dynamic Logic.* Lecture Notes in Computer Science, Vol. 68. Springer-Verlag, Berlin/New York, 1979.

[HM90] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990. A preliminary version appeared in *Proc. 3rd ACM Symposium on Principles of Distributed Computing*, 1984.

[HZ87] J. Y. Halpern and L. D. Zuck. A little knowledge goes a long way: Simple knowledge-based derivations and correctness proofs for a family of protocols. In *Proc. 6th ACM Symp. on Principles of Distributed Computing*, pages 269–280, 1987. A revised and expanded version appears as IBM Research Report RJ 5857, 1989, and will appear in *Journal of the ACM.*

[Kap77] D. Kaplan. Demonstratives. Unpublished manuscript, UCLA, 1977.

[Les91] Y. Lespérance. *A Formal Theory of Indexical Knowledge and Action.* PhD thesis, University of Toronto, 1991.

[Lew79] D. Lewis. Attitudes *de dicto* and *de se*. *Philosophical review*, 88(4):513–543, 1979.

[McC79] J. McCarthy. Ascribing mental qualities to machines. Technical Report STAN-CS-79-725, Stanford University, 1979.

[Mos92] Y. Moses. Knowledge and communication (a tutorial). In Y. Moses, editor, *Proceedings of the Fourth Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 1–14. Morgan Kaufmann, 1992.

[MW85] M. Merritt and P. Wolper. States of knowledge in cryptographic protocols (extended abstract). Unpublished, 1985.

[RK86] S. J. Rosenschein and L. P. Kaelbling. The synthesis of digital machines with provable epistemic properties. In J. Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge: Proc. 1986 Conference*, pages 83–97. Morgan Kaufmann, 1986.

Emergent Conventions in Multi-Agent Systems: initial experimental results and observations
(preliminary report)

Yoav Shoham and Moshe Tennenholtz
Robotics Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305

Abstract

Motivated by the problem of coordination among multiple agents, we are concerned with the process by which agents can make local decisions that lead to global conventions. While ultimately we are after a mathematical theory, here we summarize results of quantitative simulations that we have carried out. In each of these experiments, 100 agents start with a random strategy (in most experiments the strategy is single bit). To preclude statistical accidents, each experiment consists of a large number of trials. In each trial, a large number of times random pairs of agents meet and observe each other. The various experiments differ along a number of dimensions: the basic function by which each agent updates its strategy, the frequency with which it is allowed to update the strategy, the amount of memory it has, the amount of information exchanged in meetings, and the symmetry of interactions. We are interested in quantitative results about the efficiency of convention evolution given different properties of such dimensions. Some of our results are intuitive; others are highly surprising. Although our primary aim here is to report on the results of these experiments, we include also a few preliminary insights that we have gained from these experiments.

1 Introduction

In multi-agent AI systems, such as multi-planner systems, it is crucial that the agents agree on certain rules, in order to decrease conflicts among them and promote cooperative behavior. Without such rules, the most simple goals might become unattainable by any individual agent, or at least not efficiently attainable (just imagine driving in the absence of traffic rules). These rules strike a balance between on the one hand allowing agents sufficient freedom to achieve their goals, and on the other hand restricting them so that they don't interfere too much with one another. We assume that the importance of such rules in a multi-agent system is incontroversial, and will not waste further valuable space on motivation.

Some of these rules are social laws, designed and imposed ahead of time; traffic laws are an example. Previous papers (see [6], [7]) have investigated some aspects of this off-line design of a social law. Not all rules can be legislated in advance, however. This is either because the characteristics of the society are unknown, or because they change over time. In such cases, it is often important that the society converge on a global convention. For example, different database implementors use different formats; when databases of different types are to be consolidated, a costly process of format conversion must take place. For this reason certain conventions, namely *standards*, emerge over time.

This paper is concerned with the design of a multi-agent system in a way which promotes rapid convergence towards conventions. The only impediment to achieving this immediately is the inability to impose a standard through a central broadcast; instead, individual agents occasionally meet and observe each other, and based on this local information may update their strategy. We are interested in the effect on convergence of various update functions, and the impact of a number of variables such as the amount of memory, the frequency of update, the amount of information exchanged in pairwise interactions, and the symmetry of interactions.

Ultimately, we are after a mathematical theory. Research in a number of other fields seems relevant to our work. This includes works in mathematical economics [3], spin glass theory in physics [4], population genetics in mathematical biology [1], and leader election in distributed systems [8]. So far we have found only superficial similarities between these works and

our results. One of our immediate research goals is the development of a general framework in which to unify this other work and our own. It is tempting to adopt an existing mathematical model, but we have felt strongly that we do not understand the phenomena sufficiently to make this commitment. Instead, we have performed a large number of computer simulations, in which we tested various choices.

2 Overview of the experiments

We were interested in investigating the simplest setting in which conventions arise; it turned out that even this simple setting gives rise to quite complex behavior. We should start by being more precise about what we mean by a convention. Although the purpose of this paper is not a mathematical theory, we should mention that one elegant definition of convention is based on the definition of a *coordination problem*. That in turn can be defined in economic terms, using the notions of *payoff functions* and *Pareto optimality*. Since our aim here is different, we only mention that our setting can in fact be defined in this fashion, and refer the reader to Lewis's [5] for more details on this line of thought.

2.1 The basic setting

While the relation between our experiments and the general notion of convention is beyond the scope of the paper, the experiments themselves are easy enough to describe. All experiments involve exactly 100 agents.[1] In almost all experiments, the "convention" is the simplest one imaginable, a single bit. Each agent starts with a randomly generated bit (we have taken care to use a reliable random-bit generator). Then, in repeated iterations, a randomly selected pair of agents meet and observe each other's bit (in some experiments more information is exchanged). The number of iterations in each trial ranges from 800 to 100,000. To preclude statistical accidents, each experiment consists of a large number of trials, ranging from 800 to 4000.

In almost all experiments we assume a homogeneous society,[2] in which all agents employ the same update function. The difference among the different experiments is in this function, as well as certain additional parameters, such as the frequency with which agents are allowed to update the strategy, and the amount of memory they have. (In addition, for pragmatic reasons experiments vary somewhat in the number of trials and in the number of iterations in each trial.)

[1] This number is interesting, since it contrasts with experiments on computational ecologies by Huberman and colleagues (cf. [2]), in which tens of thousands and even millions of agents were needed in order to demonstrate phenomena of interest to them; they were not interested directly in the evolution of conventions.

[2] Some results concerning the non-homogeneous case will appear in a pending paper.

2.2 The basic strategy-update functions

We have experimented with a large number of ways in which agents update their strategies. Many of these turned out to be unilluminating; here we will describe some that did lead to insights. As will become apparent, while we have covered a lot of ground, many other update functions are possible. We return to this in the final section. In this subsection we describe the basic strategies explored; in the next section we describe a number of experiments, which modified the basic strategies and varied a few parameters.

Assume the following definitions. A meeting between two agents is called a *success* if their strategies at that time agree, and a *failure* otherwise. In any given trial, at each point in time let $good_i(j)$ denote the number of successes j experienced while using the i strategy ($i = 0, 1$) from the beginning up to that point. Similarly, let $bad_i(j)$ denote the number of failures of the i strategy for agent j up to that point. We define the frequency of success of strategy i for agent j up to that point by $prob_i(j) = \frac{good_i(j)}{good_i(j)+bad_i(j)}$. We assume that $good_i$ and bad_i are initialized to 1 for $i = 1, 2$ for all agents.

Consider the following methods for strategy update:

1. *(Internal statistics).* Adopt the strategy i with probability $\frac{prob_i(j)}{prob_i(j)+prob_{1-i}(j)}$.
2. *(External statistics).* Adopt the strategy i with probability that is the proportion of i's encountered in *other agents* so far.
3. *(Internal majority).* Adopt the strategy i if $prob_i(j) > prob_{1-i}(j)$, and remain with your current strategy in the case of equality.
4. *(External majority).* Adopt the strategy i if so far it was observed in other agents more often than the strategy $1 - i$, and remain with your current strategy in the case of equality.

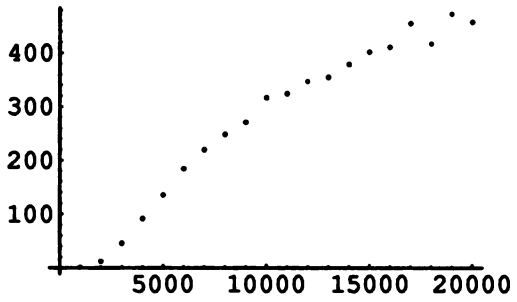


Figure 1: Internal Statistics: Idealizing Agents

We encourage the reader to speculate on these methods of update before reading on, and try to predict which of those will result in the most rapid convergence. We are not too proud to admit that our own predictions were very poorly matched by the experimental results.

3 Initial experimental results

3.1 Internal statistics

In this case we get a truly inefficient convention evolution. For example, among 4000 trials each of which included 100000 iterations, we got that in only 136 of the trials at least 80% of the agents reached a convention (the same bit/strategy).

We experimented with one modification of that approach, in which we considered *idealizing agents*, agents that ignore their previous failures. These agents adopt the 0 option at a particular point with probability $\frac{good_0(j)}{good_0(j)+good_1(j)}$. These *idealizing agents* turn out to converge on a convention much faster than those that use full internal statistics. These results are demonstrated by Figure 1. The x coordinates in Figure 1 correspond to the number of iterations in each trial, while the y coordinates correspond to the number of trials from among 800 trials in which at least 80% of the agents reached a convention.

3.2 External statistics

In this case we get (again) a truly inefficient convention evolution. For example, in no trial of 5000 iterations did we get that at least 80% of the agents agree. Even when we lowered the threshold, and asked only that at least 55% of the agents agree, we were successful in only 481 trials out of 800 trials of 5000 iterations each.

3.3 Internal majority

This option is significantly better than all options discussed so far. In 1701 from among 4000 trials of 800 iterations each we got that more than 85% of the agents reached a convention.

3.4 External majority

This option is by far the best basic option among those we have tested. In 3010 from among 4000 trials of 800 iterations each we got that more than 85% of the agents reached a convention. In 3801 from 4000 trials of 1600 iterations each, we got that more than 95% of the agents reached a convention.

3.5 Modified external statistics

The results so far might suggest a rule of thumb, namely to base one's strategy deterministically on majority measures, rather than to flip coins. We conclude this section with a result that cautions against blind application this rule. Consider again the method of external statistics, which was shown to be very ineffective. It turns out that a simple modification of it is significantly superior to the internal majority method (though still inferior to the external majority method). In this modification an agent still uses the external statistics criterion, but applies it only when it observes in another agent a strategy different from its own; in other cases it leaves its strategy unchanged. The results we obtained are that in 3403 trials from among 4000 trials of 1600 iterations each, more than 95% of the agents reached a convention. In a similar experiment in which there were only 800 iterations, in 2045 trials from among 4000 we got that 85% of the agents reached an appropriate convention. In the next section we will look more closely at the effects of frequency of update and other parameters; the reason for including this result here was to caution the reader against premature conclusions.

4 Modifications of the basic experiments

In this section we are concerned with the effects on convention emergence of various parameters such as frequency of update and amount of memory. In all of these experiments we modify the external majority update function, the one that performed best among the basic functions tested.

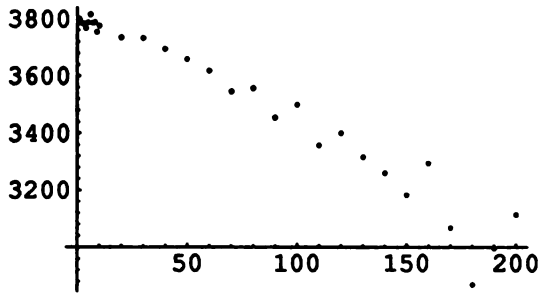


Figure 2: The effects of update frequency

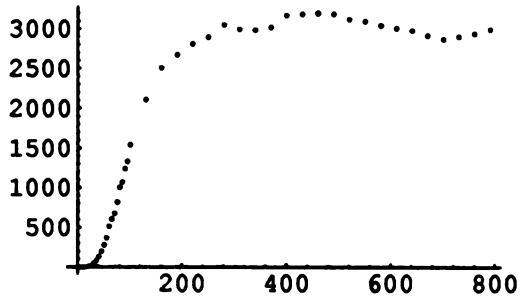


Figure 3: The effects of memory restarts

4.1 The effect of update frequency

The first type of modification we consider is concerned with update frequency. In the previous section we assumed that each agent updates[3] its behavior at each iteration. However, in some systems agents update their behavior in a less frequent manner. We investigated the effects of update frequency on the efficiency of convention evolution. We found that when the frequency of update decreases, then the efficiency of convention evolution decreases. Our results can be illustrated by Figure 2. In this figure, the x coordinates describe the distance between iterations where update is performed, while the y coordinates describe the number of trials from among 4000 trials of 1600 iterations each where more than 95% of the agents reached a convention.

4.2 The effect of memory size

Here we investigate the effects of memory size on the efficiency of convention evolution. We consider two forms of limited memory.

4.2.1 Periodic memory restarts

One type of limited memory is a memory that is restarted from time to time. When the memory is restarted, the agents' current strategies (the ones they will now start with) are not forgotten, but previous history is. This might be in particular interesting in systems which stop operating for a short while from time to time. For example, a society might be interested in a particular coordination only in some periods of the year, where agents are assumed to forget what they have exactly seen in the previous periods although they still remember their current (latest) strategy. We investigated the efficiency of convention evolution as a function of the frequency of memory restarts. We found that when the distance between iterations where the memory is restarted decreases, then the efficiency of convention evolution decreases. This can be illustrated by Figure 3. The x coordinates of this graph correspond to the distance between iterations where the memory is restarted. The y coordinates describe the number of trials from among 4000 trials of 800 iterations each, in which more than 85% of the agents reached a convention.

The reader may be tempted to treat this as an 'obvious' result; however, full memory is not always an advantage. Sections 4.2.2 and 4.3 provide some examples; here is another example. We ran an experiment in which agents restarted their memory ***always and only*** after changing their strategy. In that case the evolution of convention was even more efficient than in the case of full memory (the external majority of section 3); in 3298 from among 4000 trials of 800 iterations each, more than 85% of the agents reached a convention.

4.2.2 Limited memory windows

A more continuous form of limited memory is one in which each agent at each time keeps a limited window into its past experience, and bases the majority rule on only that window. We have considered two forms of windows, one in which it remembers the last n iterations in which it participated in a meeting, and another in which the agent remembers the last n iterations, regardless of whether it participated in a meeting in those.

Our results of these two experiments are illustrated in Figures 4 and 5, respectively. In both of these figures the x coordinates describe the size of memory window, and the y coordinates correspond to the number of trials from among 4000 trials of 800 iterations each, in which more than 85% of the agents

[3] By 'update' we mean the application of the update function; the result need not be a change in strategy.

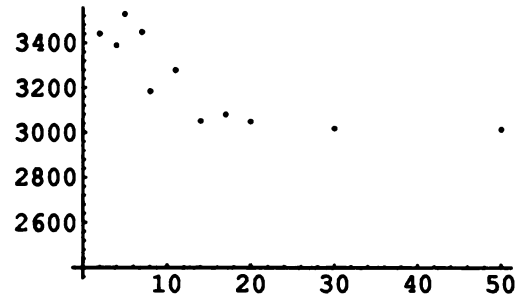


Figure 4: Limited Memory (latest observations)

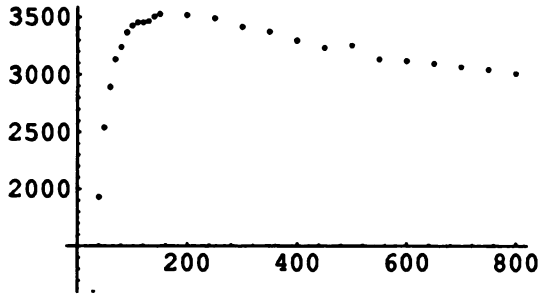


Figure 5: Limited Memory (latest iterations)

reached a convention. Note that, somewhat surprisingly, in both cases it pays to forget, though some minimal memory is essential (in the first case this minimum is in fact equal to 2 iterations, and therefore this can be seen more easily in the second case).

4.3 Co-varying memory size and update frequency

We have so far varied update frequency and memory independently; we now show that these two parameters interact. Consider the results from section 4.1, where we showed that the rate of convention evolution is a monotonic increasing function of update frequency. We now show that decreasing memory blocks the degradation of convergence with the decrease in update frequency. Specifically, in this experiment we adopted the memory-restart model, and varied together the memory-restart frequency and the update frequency. The general result we obtained is that when update becomes infrequent (there is a long delay between strategy updates), then it is better to restart the memory from time to time than to rely on the whole memory. Our results are illustrated in Figure 6. The x coordinate of this figure corresponds to the update frequency, which is equal to the number

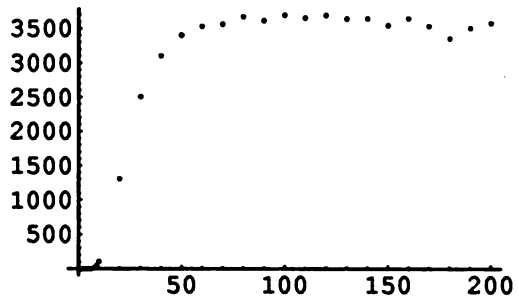


Figure 6: The case where update frequency = memory restart frequency

of iterations between consecutive memory restarts. That is, in this case, we had a single interval which served both as the update frequency and the memory restart frequency. The y coordinates correspond to the number of trials from among 4000 trials of 1600 iterations each in which 95% of the agents reached a convention. It is illuminating to compare Figure 6 to Figure 2 (where full memory is assumed); when the update frequency drops below about 100 iterations, it becomes better to use the statistics of only the last window than to rely on the entire history.

4.4 Communicating past history

Up until now we have assumed that agents do not communicate any information to other agents except their current strategy. In this experiment we test the effect of more elaborate communication. Specifically, we assume that when agents meet, they exchange their full prior histories. We assume that agents give similar weight to their experience and to that of others: after a communication act each agent updates its history of observations according to the average of its experience and that of the other agent (of course, the statistics of the two agents will diverge again in the future). To make things interesting, we do not assume that *any* pair of agents engage in this full disclosure when they meet. Rather, we define the notion of *extroversion radius*, which crudely captures the fact that agents tend to confide in only some other agents. Specifically, we number the agents $1, 2, \ldots, n$, and when the radius of extroversion is r it means that agent i exchanges full information in its encounters with agent j iff $j \in [i - r..i + r](mod\ n)$; with other agents the exchange is only the current strategy.

We investigated the relation between the radius of extroversion and the efficiency of convention evolution. We found that a small radius of extroversion increases the efficiency of convention evolution in a

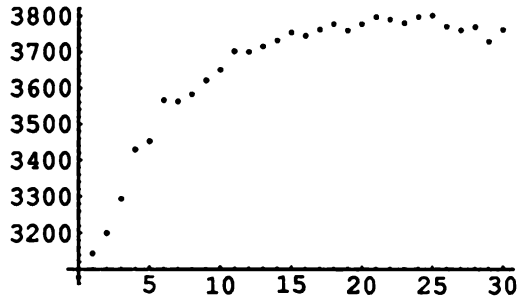


Figure 7: The effects of communication

significant manner relative to the single bit exchange case. However, a big radius of extroversion does not provide additional help, and thus not worth the additional cost. We demonstrate our results in Figure 7. The x coordinate in that figure corresponds to the extroversion radius. The y coordinate describes the number of successful trials (more than 85% of the agents reached a convention) from among 4000 trials of 800 iterations each.

4.5 Asymmetric interactions

In all of the previous results we assumed that when agents meet then they are able to observe each other's strategy. The other symmetric case where both agents who meet cannot observe each other is vacuous. However, an interesting complementary case is the asymmetric one, when only one of the agents who meet can observe the other's strategy. This asymmetric case seems at first much similar to the symmetric one. In fact for most types of behaviors the efficiency of convention evolution will simply decrease by a factor of 2. However, we can show interesting qualitative differences between the symmetric and asymmetric case.

For example, the trivial option of memory restart at each iteration will lead to nothing in the symmetric case (the situation will remain as the initial one) but will converge to an appropriate convention in the asymmetric one. In some cases the fact that there are less updates in the asymmetric case will lead to qualitative differences between the symmetric and asymmetric cases when the number of iterations in each trial is relatively small. Consider for example an update function where an agent chooses the option 0/1 if in the two latest iterations it was involved with it observed (in other agents) the option 0/1 respectively (and otherwise it remains with its current strategy). If we look at the result of applying this update function along 800 iterations in the symmetric and asymmetric cases, we get: This update function gives extremely good results in the symmetric case (much better than relying on full memory), but it is worse than relying on full memory in the asymmetric case. We will expand on this in a longer version of this paper.

4.6 More complicated decisions

Up until now we assumed that the agents must decide on a binary value; a natural extension is concerned with more complicated decisions. What happens if the agents must agree on one option among more than two available options, that is, on something more complicated than a bit? How does the number of options (potential conventions) effect the efficiency of convention evolution?[4]

Our general results are as follows. What we find is that adding more potential conventions decreases the efficiency of convention evolution in a less than logarithmic fashion. In addition we find that the absolute amount of success in convention evolution decreases in less than logarithmic fashion: In order that the number of successes of convention evolution will decrease by factor of 2, we need to increase the number of potential conventions by a factor of more than 4, and in order to decrease it by a factor of 3 we need to increase the number of potential conventions by a factor of more than 8.[5] Intuitively speaking, our results point to the following encouraging fact: the efficiency of convention evolution is not affected too badly by an increase in the number of potential conventions.

Our specific results are illustrated in Figure 8. The x coordinate describes the number of potential conventions, while the y coordinate describes the number of successful trials (more than 85% reached a convention) from among 4000 trials of 800 iterations each.

5 Conclusions

In this paper we shared with the reader results and observations about the efficiency of convention evolution. The understanding of the efficiency of convention evolution is an important step for the design of non-trivial multi-agent systems. Our current results shed light on the effects of different elementary agent

[4] We still assume the external majority update rule, but now the agent adopts a strategy (option) that was observed more often than all of the other strategies.

[5] We have verified these basic results also in the case of limited memory.

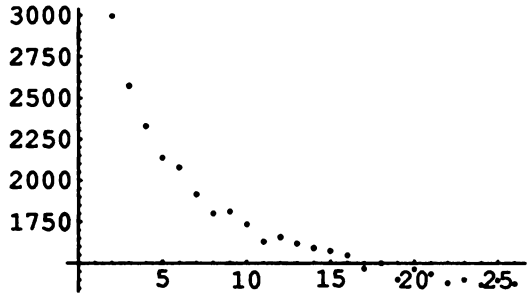


Figure 8: The effects of the number of potential conventions

behaviors and system characteristics on the efficiency of convention evolution. Some of these results can be used directly when designing artificial systems, while others can be used in order to explain or predict the behavior of societies and as a result to enable useful participation in them.

There is still much to be done. First, more experiments are needed in order to explore additional interesting phenomena concerning the efficiency of convention evolution. We would like to test our experimental results against certain real-world phenomena, such as fashion fads. Finally, as mentioned before, the desirable goal is a mathematical theory which explains results such as those discussed above, and, better yet, predicts new ones.

References

[1] L. Altenberg and M. W. Feldman. Selection, Generalized Transmission, and the Evolution of Modifier Genes. I. The reduction principle. *Genetics*, pages 559–572, November 1987.

[2] Bernardo A. Huberman and Tad Hogg. The Behavior of Computational Ecologies. In Bernardo A. Huberman, editor, *The Ecology of Computation*. Elsevier Science, 1988.

[3] M. Kandori, G. Mailath, and R. Rob. Learning, Mutation and Long Equilibria in Games. Mimeo. University of Pennsylvania, 1991.

[4] R. Kinderman and S. L. Snell. *Markov Random Fields and their Applications*. American Mathematical Society, 1980.

[5] David Lewis. *Convention, A Philosophical Study*. Harvard University Press, 1969.

[6] Y. Moses and M. Tennenholtz. On Computational Aspects of Artificial Social Systems. In *the Proceedings of DAI-92*, 1992.

[7] Y. Shoham and M. Tennenholtz. On the Synthesis of Useful Social Laws for Artificial Agent Societies. In *Proc. of AAAI-92*, pages 276–281, 1992.

[8] M. Yamashita and T. Kameda. Computing on Anonymous Networks. In *Proc. 7th ACM Symp. on Principles of Distributed Computing*, pages 117–130, 1988.

Knowledge Representation Requirements for Description-based Communication*

Anthony S. Maida

The Center for Advanced Computer Studies
The University of Southwestern Louisiana
Lafayette, LA 70504-4330

Abstract

This paper describes a benchmark problem which must be solved if we are to construct autonomous agents which are able to engage in description-based communication. The problem is framed in terms of a communication scenario called the *divided reference scenario*. We describe representations of the communicating agents' mental states at critical points in this scenario. We also discuss the inferential requirements needed to allow an agent to autonomously construct these representations.

1 Introduction

This paper is concerned with designing a transaction language to allow autonomous agents to exchange information about newly discovered objects [23,26]. As an agent explores a domain, the agent may be unaware of the existence of an object until the agent actually encounters the object. When the agent does encounter the object, the object will not have a standard name. Consequently, it is not obvious how the agent would be able to tell another agent about the object.

An agent could attempt to dynamically generate standard names by linking them to the object's location. For instance, if an agent always knew its own location, then it would know the location of any object that it approached and could therefore generate a description of the form "the object at location x, y, z." There are two complications with this idea. First, more than one object could have the same location, such as a container and its contents. Second, objects can change location. One could consider adding a time parameter as part of the object's name such as "the object at location x, y, z at time t." But now, since an object could have different locations at different times, it would have different descriptions at different times, and agents would be faced with the problem of determining when different descriptions applied to the same object, leading us into the complexities of description-based communication (which is the topic of this paper).

Another way to enable an agent to refer to an unnamed object could be by "pointing" in some fashion. The agent need not literally point with a finger. All that is needed is some way for the agent to highlight the object so that the receiving agent can sense which object is being highlighted. This method appears to require some type of co-presense among the agents; the sending agent cannot usefully point to an object which the receiving agent cannot see. Possibly, there are ways around this limitation. For instance, the sending agent might be able to transmit an image of the scene to the receiving agent and then highlight the object being referred to by using an electronic cursor.

This paper investigates a description-based approach to solving this problem. The approach is suggested by drawing an analogy to a similar communication problem in natural language [9,1,19]. In natural language, objects which do not have names can be referred to by use of *descriptions*. Roughly, a description is a specification of a set of properties that an object has which distinguishes it from other objects in the domain. Given an object to which an agent intends to refer, a description is satisfactory if the properties it specifies uniquely match the object relative to the domain of discourse. A description is faulty otherwise.

This paper analyzes some of the knowledge representation (KR) requirements needed to support description-based communication between autonomous agents. For concreteness let us assume a domain with the following characteristics.

1. The domain has a finite number of objects.
2. Multiple agents, who have standard names,[1] explore the domain.

*The author can be reached electronically at maida@cacs.usl.edu.

[1] Grove and Halpern [14], in an effort to weaken this

3. Agents, for the most part, acquire knowledge of objects in the domain by discovering them as they explore the domain.
4. Agents may wish to exchange information about objects that they encounter.
5. Objects do not have standard names.
6. Objects have properties and the properties have standard names.
7. No two objects in the domain have exactly the same properties.

This type of domain presents a realistic model in which to study description-based communication. In this domain, simpler forms of reference are not possible because objects do not have standard names. On the other hand, description-based reference seems feasible. If agents had complete knowledge of the objects and their properties in this domain, then by assumptions 6 and 7, they could use descriptions (specifications of sets of properties) to uniquely signal which objects they want to refer to. Since agents do not have complete knowledge of domain objects and their properties, an agent cannot be sure that a description it generates is satisfactory. Because of this uncertainty, there is always potential for a reference identification failure (RIF).

A RIF occurs when one agent, the sending agent, uses a description intending to refer to some entity, but the receiving agent is unable to identify, or misidentifies, the sending agent's intended referent. Since RIF's appear to be inherent to description-based communication, we shall focus on requirements for the diagnosis of, and recovery from, RIF's.

2 Basic Considerations

In order to implement a fully functional description-based transaction language, one must develop algorithms to detect and recover from RIF's whenever they occur. Both detection of, and recovery from, RIF's is difficult. RIF's in autonomous-agent communication, and in human communication, cannot always be detected directly. A RIF may manifest itself as some other kind of anomaly which is subsequently diagnosed as a RIF [13].

The cause of a RIF, once diagnosed, is always a misconception (c.f., section 3.2). Consequently, to recover from a RIF, an agent must correct the misconception and resume communication from that point on.

assumption, introduce explicit names for agents into the semantic domain of a possible worlds logic. They state "quantifying-in over agents is replaced by quantifying-in over names (p. 309)." This is a standard device in syntactic logics [15,25,22] of belief. Konolige's bullet operator is a related device [12, p. 217].

2.1 The Divided Reference Scenario

This section describes a benchmark problem which must be solved if we are to build autonomous agents that engage in description-based communication. For concreteness, we use a particular domain that has the seven characteristics listed in section 1. Part of the domain is shown in Figure 1. The initial state of the domain is described below.

> There are two agents, **Agent1** and **Agent2**, initially outside of a barn. The barn has two entrances — a front entrance and a rear entrance. Inside of the barn, there is a tractor near each entrance. The barn is large (or perhaps, L-shaped) and if an agent sees a tractor at one entrance, then the agent will be unable to see the tractor at the other entrance. Since the agents are outside of the barn, they have not seen the tractors and do not know of their existence. The tractor at the rear entrance of the barn is obviously broken and an agent will notice this if it sees the tractor.

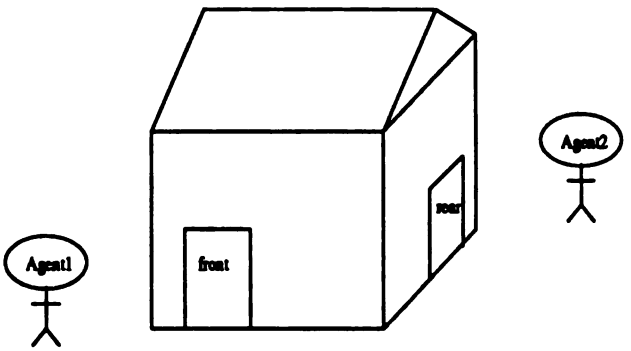


Figure 1: A Barn with **Agent1** and **Agent2** near the Front and Rear Entrances, Respectively.

The following scenario consists of five events and leads to a RIF which is difficult to diagnose. This kind of scenario could potentially occur in any domain that has the seven characteristics described in the introduction. Therefore, the diagnosis problem for RIF's is a general problem inherent to description-based communication.

Event 1: **Agent1** enters the barn using the front entrance, sees the tractor at the front of the barn, and then exits the barn.

Event 2: **Agent1** tells **Agent2** that *there is a tractor in the barn.*

Event 3: **Agent2** enters the barn using the rear entrance, sees the broken tractor near the rear entrance of the barn, assumes that this is the tractor that **Agent1** referred to, and then exits the barn.

Event 4: **Agent2** says to **Agent1** that *the tractor is broken.*

Event 5: Agent1 notices an apparent discrepancy between his own beliefs and those of Agent2.

In the third event, **Agent2** misrecognizes one tractor as being the same as a different tractor. At this point, **Agent2** acquires an existential misconception. The fourth event describes an occurrence of divided reference; **Agent2** is unwittingly refering to two objects. Finally, in the fifth event, there is the first indication of a communication anomaly. **Agent1** notices an *apparent* discrepancy in belief about whether 'the tractor' is broken.

2.2 The Problem

How can we design **Agent1** so that it can recover from the RIF in the scenario just described?

2.2.1 The Detection Problem

First, let us describe the detection problem. The relevant causal events in the scenario are as follows.

1. **Agent2 sees an object that Agent1 does not know about.**
2. **Agent2 makes a recognition error, thereby acquiring an existential misconception.**
3. **Agent2 uses a faulty description (divided reference).**
4. **Agent1 notices an apparent belief discrepancy.**

But **Agent1**, not being psychic, must determine the cause of the apparent belief discrepancy. From **Agent1**'s perspective, the discrepancy could be caused by one of three states of affairs listed below. Let the notation $P(x)$ stand for the proposition *x is broken*.

1. **Agent1 is correct in not believing $P(x)$ and Agent2 is incorrect in believing $P(x)$.**
2. **Agent2 is correct in believing $P(x)$ and Agent1 is incorrect not believing $P(x)$.**
3. **Agent1 and Agent2 are talking about different things.**

The three possible causes are shown in Figure 2 where the arrows indicate a possible causal relationship. In the first two cases, the apparent belief discrepancy is really caused by an actual belief discrepancy. In the third case, at least one of the agents has a misconception (**Agent2** in this example) which causes a RIF, which in turn causes the apparent belief discrepancy. Since the misconceptions are existential misconceptions [21], they cannot be characterized as belief discrepancies where one agent believes $P(x)$ and the other does not.

2.2.2 Existential Misconceptions

Hirst [16] offers an informative taxonomy of varieties of states of existence that must be describable in a KR formalism for natural language applications. However, Hirst's taxonomy does not include circumstances

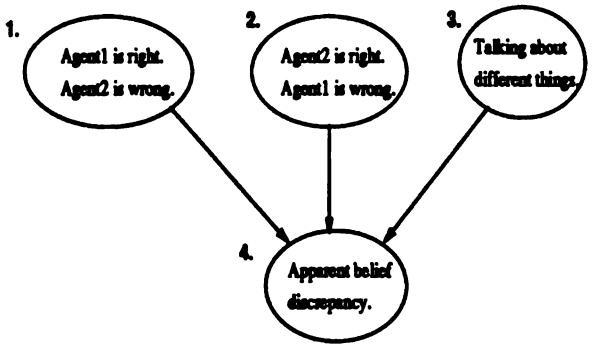


Figure 2: Three Possible Causes of an Apparent Belief Discrepancy.

where an agent has incomplete knowledge of an object's existence [21]. I call these situations *existential misconceptions*. Here we note two kinds of existential misconceptions that are relevant to description-based communication.

The first type of existential misconception is *compression*. This occurs when an agent unwittingly believes that two distinct objects are one. In representational terms, this causes the agent to compress the information for two objects onto one mental representation. **Agent2** suffers from a compression misconception.

The second type of existential misconception is *dispersion*. This occurs when an agent thinks there are two or more objects when in reality there is only one object. In representational terms, the agent unwittingly disperses the information for the one real-world object among more than one mental representation. For example, the Greek astronomers did not realize that *Hesperous* and *Phosphorous* were the same object — the planet Venus. For another, biologists, for some years, did not realize that serotonin and enteramine were the same compound — 5–hydroxitryptamine.

Compression and dispersion misconceptions are formally defined in terms of an agent's internal representations in section 4.2.

2.2.3 KR Requirements for Detection and Diagnosis

One of the KR requirements to allow **Agent1** to diagnose the cause of the apparent belief discrepancy is that **Agent1** must be able to explicitly represent each of the three possible causes shown in Figure 2. The first two possible causes are actual belief discrepancies and are not difficult to represent. The third possible cause — the actual cause in this example — is **Agent2** being in the state of having an existential misconception. This is more difficult for **Agent1** to explicitly represent [21]. Sections 3 and 4 of the paper develops theory and notation needed to do this.

2.2.4 Recovery

The root cause of the RIF in the divided reference scenario is **Agent2**'s existential misconception. One way to recover from a RIF after the misconception is diagnosed is for **Agent1** to give **Agent2** information which can be used to correct the misconception, and then to restart the original reference process after that.

2.2.5 Plan of the Paper

This paper is organized around constructing representations of **Agent1**'s model of **Agent2** at three crucial points in the divided reference scenario. First, we represent **Agent1**'s model of **Agent2** at the time the discrepancy is discovered but before it is diagnosed. Second, we represent **Agent1**'s model of **Agent2** after diagnosis. Third, we represent **Agent1**'s model of **Agent2** after recovery. The theory underlying these representations is presented in section 3. The representations themselves are presented in section 4. Finally, the inferential requirements needed for an agent to autonomously construct the representations are presented in section 5.

3 Reference and Internal Representation

This section discusses the relationship between the reference process and internal representation in the context of the divided reference scenario. Figure 3 schematically depicts a portion of the agent's mental states after the events 1, 2, and 3 take place. The figure is incomplete in that it does not show each agent as having a model of the other agent's mental state. This deficiency will be addressed later.

For now, let us explain the figure. After **Agent1** enters the barn through the front entrance and observes the tractor at that entrance, the agent will create a symbol (which we shall notate as "S1") to represent that tractor. Arrow 1 indicates that **Agent1** uses this symbol to represent the tractor at the front entrance. The arrow is not part of the agent's knowledge base, but is part of the vocabulary of an external language to allow an observer to describe the agent's mental state.

After **Agent1** tells **Agent2** that there is a tractor in the barn, **Agent2** will construct a representation (which we shall notate "S2") for the tractor that **Agent1** referred to. This is arrow 2 shown in Figure 3. We use a dashed arrow to indicate that the nature of the representation relationship between "S2" and the tractor is less direct than the relationship between "S1" and the tractor.

After **Agent2** enters the barn from the rear entrance, **Agent2** will construct a representation (which we shall notate as "S3") for the tractor at the rear of the barn. This representation relationship is indicated by arrow 3. Arrow 3 is a solid arrow, just as arrow 1 is, because the nature of the representation relationship

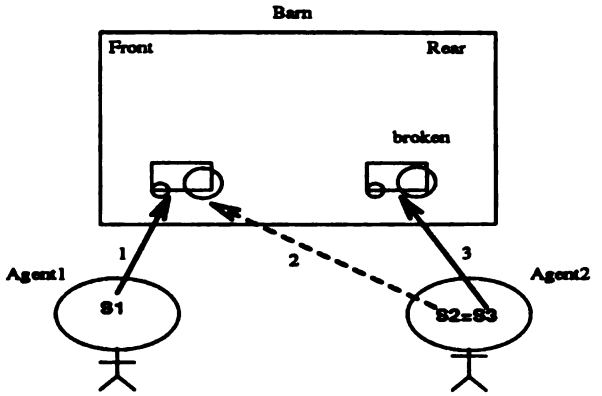


Figure 3: Mental Symbols and the Representation Relationship.

between "S3" and the tractor at the rear of the barn is one of direct perception. There is an equality sign between "S2" and "S3" to indicate that the agent treats the distinct representations as representing the same object.

3.1 Individuating Sets

Appelt and Kronfeld [2] have applied the term *individual object representations* (IOR's) to representations such as "S1", "S2", and "S3". In their terminology, "S1" and "S3" are discourse IOR's and "S2" is a perceptual IOR. Discourse IOR's are created by information acquisition through a discourse process and perceptual IOR's are created by information acquisition via a perceptual process.

It is natural to model an agent's inference procedure as exhaustively computing inferences involving identity, symmetry, and transitivity of equality. In this circumstance, ground terms in the agent's knowledge base can be partitioned into equivalence classes based on the agent's equality relation. Two ground terms are in the same equivalence class for an agent if the agent believes that the expressions are equal. These equivalence classes are called *e-classes* in [21] and *individuating sets* in [2]. The latter term is more informative about the purpose of the equivalence class from the viewpoint of the agent. In particular, the agent treats the class as a representational unit that stands for a unique individual. Because of these considerations, we can think of the individuating set $\{S2, S3\}$ as a structure which is at the tail of arrows 2 and 3.

3.1.1 Divided Reference

From the previous discussion, it is clear that **Agent2** is using the single individuating set $\{S2, S3\}$ to represent two distinct entities in slightly different ways. Consequently, when **Agent2** uses the description *the tractor*, there is a case of unwitting divided reference. **Agent2**

is intending to refer to the entity that **Agent1** referred to during the second event and **Agent2** is also intending to refer to the entity **Agent2** saw at the rear of the barn. Once this divided-reference event occurs, the RIF is certain to occur because there is not a unique intended referent for **Agent2**'s description. The root cause of the divided-reference event is **Agent2**'s existential misconception.

In the Appelt/Kronfeld model, their referring schema is undefined for this circumstance. For them the symbol "S2" is an IOR that denotes tractor1 and "S3" is an IOR that denotes tractor2. Since "S2" and "S3" are elements of the same individuating set, but their denotations are discrepant, the set has no referent [2, p. 643]. Therefore, their referring schema is undefined for this situation and cannot be applied. **Agent2**, in their analysis, is in fact not referring at all! The basic problem is that the Appelt/Kronfeld framework cannot describe agents who have compression misconceptions.

Regardless of whether one calls it "reference," the divided reference scenario can and will happen. If one wants to build a description-based communication language for autonomous agents that works, one must have a method to model this scenario.

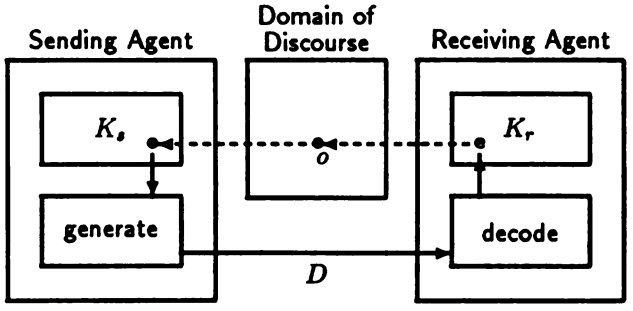


Figure 4: A Process Model for Reference and Reference Identification.

3.2 Causes of RIF's

Figure 4 depicts a process model for definite reference [7]. Analysis of the model shows that RIF's are caused by at least one of the communicating agents possessing inaccurate or incomplete knowledge of the domain of discourse.

The figure shows a domain of discourse, a sending agent, and a receiving agent. The domain of discourse is shown in the middle of the figure. The arrows in the figure indicate the directionality of the information flow. The domain of discourse contains an object, o, that is the sending agent's intended referent. Each agent uses its own knowledge base to represent the domain of discourse (K_s and K_r). The sending agent uses an encoding process to construct a description for o. This encoding process takes as input an entity in the sending agent's model, K_s, that represents o and produces a description, D, that uniquely matches o relative to K_s. D is then sent to the receiving agent who then applies a decoding process to D which operates relative to the receiving agent's model K_r. The decoding process in the receiving agent, if successful, will flag an entity in K_r as representing o. If the receiving agent does not flag a unique representation for o, then there is a RIF. It is reasonable to assume that the description generating and decoding processes in the sending and receiving agents have been debugged.[2] Therefore, with respect to this model, the causes of RIF's are due to either one or both of the agents having inaccurate or incomplete knowledge of the domain of discourse.

In general, if a RIF occurs, it is caused by a misconception in either the sending agent or receiving agent. Not all misconceptions leading to RIF's are existential misconceptions, but these are the ones that are difficult to represent. An agent cannot understand the cause unless it can explicitly represent the content of the misconception.

3.3 A Distinction Between Representation and Denotation

It is tempting to model the representation relationship in terms of denotation. For instance, Nilsson [24, p. 37] attempts to formalize the concept of representation by directly linking it to the denotational apparatus of predicate calculus. Similarly, Appelt and Kronfeld [2] speak of the denotation of IOR's and individuating sets. Unfortunately, the arrows in Figure 3 depict the *representation* relationship and not the *denotation* relationship. With respect to individuating sets, the argument is clear.

1. Mental tokens can simultaneously represent more than one object (e.g., the individuating set $\{S2, S3\}$ simultaneously represents tractor1 and tractor2 for **Agent2**).
2. Symbols in a language (mental or otherwise) can formally denote at most one object.
3. Therefore, denotation is not a natural way to formalize representation.

As we mentioned, Appelt and Kronfeld would argue that the type of individuating set exemplified by $\{S2, S3\}$ has no denotation. The present paper agrees

[2]This would not be true if either the sending or receiving agent were human. Humans are obviously error prone when they generate descriptions under real-time constraints [9,8]. Similarly, it is easy to imagine humans making errors when decoding descriptions. I also do not want to imply that the description generation process is simple [10].

but also points out that the individuating set does in fact *represent* two distinct entities.

3.3.1 IOR's

With respect to IOR's, Appelt and Kronfeld claim that IOR's like "S2" and "S3" have denotations. In particular, they would say that "S2" denotes tractor1 and "S3" denotes tractor2 for Agent2. With respect to perceptual IOR's, such as "S3", the claim seems initially plausible but it requires that the agent be guaranteed to generate a distinct perceptual IOR for each distinct object it observes. In this example, we would have to guarantee that Agent2 never generates "S3" when it is looking at an object that is not tractor2. Because distinct objects may appear identical in various situations (e.g., identical pennies or identical twins), it seems unlikely that a perceptual system could ever be constructed that is guaranteed to always generate distinct IOR's for distinct object, unless it always generated a new IOR for anything that it observed.

With respect to discourse IOR's such as "S2", the claim that these IOR's have denotations is less plausible. Consider the following example. Suppose, after the third event, that Agent2 says to a third agent, who has knowledge of neither tractor1 or tractor2, that *the tractor is broken*. We have already established that Agent2's act of reference is divided. Therefore, if the third agent were to construct a discourse IOR, intending to denote what Agent2 referred to, it would not denote a unique object.

3.3.2 The Individuation Principle

The Appelt/Kronfeld model operates in a framework where denotation is equated with representation and that is why their model breaks down in circumstances of divided reference. The philosophical foundations of their model are discussed in Kronfeld [19, p. 141] where he explicitly describes the reference process in terms of the construct of denotation.

> ...the speaker has a mental representation denoting what he believes to be a particular object, and he intends the hearer to come to have a representation denoting the same object ... (p. 141)

Kronfeld [19] goes on to be even more explicit.

> Reference, according to this view, is by virtue of *denotation* — the crucial relation between mental representations and objects. (p. 141)

The view that internal representations denote objects is termed *the individuation principle*. Kronfeld [19, p. 41] attempts to prove that the individuation principle must be true by making reference to what he calls the *trivial principle*. The trivial principle states that for a particular belief, an agent either holds the belief or does not hold the belief. The problem with the trivial principle is that it is based on a folk psychological conception of belief ascription. This conception breaks down in circumstances where agents have existential misconceptions. For comparison, consider the statement: *Either the U.S. won the Gulf war or the U.S. did not win the Gulf war.* The statement is clearly debatable, but if the trivial principle were true with respect to winning a war, then the statement would have to be true.

With respect to belief ascription, Kripke's [18] puzzle about belief presents a direct challenge to the trivial principle. Kripke describes a person named "Pierre" who, as a child in France, read about London and came to believe that London is pretty. However, the books that Pierre read were in French and Pierre refers to London by the name "Londres." Pierre will respond affirmatively to the question, when posed in French, "Is London pretty?" As an adult, Pierre learns English and lives in London, but never comes to realize that the London he lives in is the "Londres" he read about as a child. Pierre lives in an unattractive part of London and would respond negatively to the question, posed in English, "Is London pretty?"

Kripke invokes two principles, one of *disquotation* and the other of *translation*. By the disquotation principle, if Pierre truthfully answers affirmatively to the question "Is London pretty?" then Pierre believes that London is pretty. From the translation principle, it follows that the semantic content of a question is unchanged if it is accurately translated from one language to another. Because of these principles, we must interpret Pierre's responses as indicating both that Pierre believes that London is pretty and that Pierre does not believe that London is pretty. Since this conclusion seems to be inconsistent, it appears that one of the principles (translation or disquotation) leading to the conclusion is be false. Alternatively, the trivial principle could be false.

3.3.3 Summary

In summary, the Appelt/Kronfeld model attributes the use of two kinds of representational structures to agents, namely, IOR's and individuating sets. The present paper uses those structures but gives them a representation-based semantics. The original Appelt/Kronfeld model uses a denotation-based semantics. For reasons stated in section 3.3.2, the claim that the denotation-based semantics must be used is subject to challenge. For reasons stated in section 3.3.1, this semantics presents an inaccurate model commonsense representation. For purposes of comparison, let me state the two alternatives below.

> **Denotation-based Framework.** Agents have beliefs by maintaining expressions in a formal

> language in their knowledge bases. The ground terms which appear in these knowledge bases denote ordinary[3] individuals.

I advocate the *representation-based framework* [17]. This framework uses the metaphor of commonsense representation. The representation-based framework makes weaker claims about the nature of representations in an agent than does the denotation-based framework. The denotation-based framework appears to make false claims to which the representation-based framework is uncommitted.

> **Representation-based Framework.** Agents have beliefs by maintaining expressions in a formal language in their knowledge bases. The ground terms which appear in these knowledge bases represent ordinary individuals.

3.3.4 Significance of the Distinction

The choice of theoretical framework affects the choice of which predicates are used in the external representation language to model the arrows shown in Figure 3. Let the expression $D_A(o,s)$ mean that agent A uses the internal symbol s to denote object o. For comparison, let the expression $R_A(o,s)$ mean that agent A uses the internal symbol s to represent object o. The denotation predicate must satisfy the following property.

$$\forall o_1 \forall o_2 \forall s \; D_A(o_1,s) \wedge D_A(o_2,s) \Rightarrow o_1 = o_2$$

In other words, an agent cannot use one internal symbol, or IOR, to denote two different entities. A representation predicate does not have this restriction. The trade-off is that the representation statement makes a weaker statement. It is important to study the utility of these weaker statements. We may need different kinds of representation predicates to describe different kinds of representation relationships. In what follows, I use a quotational logic [25,15,21,22] with a denotational semantics and I use two kinds of representation predicates.

4 Formally Representing Mental States

This section characterizes the KR requirements for the RIF diagnosis problem in circumstances of divided reference. To do this we need to represent `Agent1`'s mental state at two different points. First, we need to represent `Agent1`'s mental state at the end of the fifth event when the apparent belief discrepancy is first detected. Next we need to represent `Agent1`'s mental state after `Agent1` has correctly diagnosed the cause of the discrepancy.

Once the KR requirements are stated, we can specify the inferential aspects of the diagnosis problem.

4.1 Notational Conventions

We shall describe the mental states of `Agent1` and `Agent2` using an external language. This language is predicate calculus with quotation. Sentences and terms are added to the conceptualization [12, p. 9] and the predicate calculus syntax is augmented with a consistent naming convention for the sentences and terms that have been added to the conceptualization [11, p. 76]. To do this we will use a standard quotation operator. In recognition of the Appelt/Kronfeld [2] distinction between discourse IOR's and perceptual IOR's, we introduce generic processes of information acquisition into the conceptualization — namely, the processes of *direct perception* and *description decoding*.

In order to make statements about individuating sets, we introduce a function symbol "`set`" that is used to create terms that denote sets. The expression "`(set 'S2 'S3)`" denotes the set $\{S2, S3\}$ that contains the two symbols "`S2`" and "`S3`". As syntactic sugar we write $\{S2, S3\}$ instead of the expression "`(set 'S2 'S3)`".

The formulas in this paper assume that we, as God-like observers, know which symbols the agents are using. In the absence of this knowledge, we would need to modify the formulas to replace the particular symbols with existentially quantified variables. However, when we model an agent as it represents a second agent's belief state, we shall not assume that the first agent knows which symbols the second agent uses. Also, if a symbol is a standard name for an entity, we assume that all agents use the symbol to represent the entity.

Let us introduce two representation predicate symbols: $\texttt{rep}_\texttt{d}$ and $\texttt{rep}_\texttt{c}$. These stand for representation in a *direct sense* and representation in a *collective sense*, respectively. To say that an agent uses an IOR to represent an object as the result of some process of information acquisition (such as perception), we use the 4-place predicate symbol "$\texttt{rep}_\texttt{d}$". Expression {1} states that `Agent1` uses the IOR "`S1`" to represent `tractor1` as a result of direct perception.

{1} $(\texttt{rep}_\texttt{d}$ `Agent1 tractor1 'S1 direct-perc`$)$

It is also useful to write that `Agent2` uses the individuating set $\{S2, S3\}$ to collectively represent `tractor1` and also `tractor2` as shown in expressions {2} and {3}.

{2} $(\texttt{rep}_\texttt{c}$ `Agent2 tractor1` $\{S2, S3\})$

{3} $(\texttt{rep}_\texttt{c}$ `Agent2 tractor2` $\{S2, S3\})$

The intuition behind this terminology is that the set

[3] By "ordinary" individual, I mean something mundane like Fido, the dog, rather than some subtle semantic entity like the set of all situations in which Fido occurs.

of IOR's in an individuating set collectively represent the object in question. Table 1 shows constant symbols used and their denotations.

Table 1: The Meaning of the Basic Constant Symbols in the External Language.

Symbol	*Denotation*
`Agent1`	Standard name for the first agent.
`Agent2`	Standard name for the second agent.
`tractor1`	The tractor at front of barn (not a standard name).
`tractor2`	The tractor at rear of barn (not a standard name).
`direct-perc`	Standard name for the process of direct perception.
`des-decode`	Standard name for the process of description decoding.

The 2–place predicate symbol "`bel`" states that a formula resides in an agent's knowledge base. Formulas are named by prefixing a single quote mark to them. This is an abbreviation for treating formulas as lists and using the function symbol "`list`" to create terms that denote lists. For instance, the expression "`(list 'A 'B)`", is abbreviated as "`'(A B)`" and denotes the list "`(A B)`".

4.2 Formal Definitions for Existential Misconceptions

We now define compression misconceptions in terms of the relationship between an agent's internal representations and reality. In representational terms, if an agent uses one e-class to represent two distinct entities, then the agent suffers from a compression misconception.

$$\{4\}\ \forall a\ \exists e\ \exists o_1\ \exists o_2\ (\text{rep}_C\ a\ o_1\ e) \wedge (\text{rep}_C\ a\ o_2\ e) \wedge o_1 \neq o_2 \Leftrightarrow (\text{compress}\ a\ o_1\ o_2\ e)$$

In the above formula, the letters a, e, and o stand for agent, e-class, and object, respectively.

Dispersion misconceptions are defined in a similar manner. In representational terms, if an agent uses two distinct e-classes, or individuating sets, to represent the same object, then the agent suffers from a dispersion misconception.

$$\{5\}\ \forall a\ \exists e_1\ \exists e_2\ \exists o\ (\text{rep}_C\ a\ o\ e_1) \wedge (\text{rep}_C\ a\ o\ e_2) \wedge e_1 \neq e_2 \Leftrightarrow (\text{disperse}\ a\ o\ e_1\ e_2)$$

Algorithms to test for and classify existential misconceptions are given in [21].

4.3 Representing Mental State

Let us describe `Agent2`'s mental state both in the presense of the compression misconception and what it would be if it were actually reflecting the state of reality.

At the end of the third event in the divided reference scenario, `Agent2` uses the IOR "`S2`" to represent `tractor1` as a result of description decoding. Similarly, `Agent2` uses the IOR "`S3`" to represent `tractor2` as a result of direct perception. As external observers, we can describe this using expressions {6} and {7} below.

{6} $(\text{rep}_d$ `Agent2 tractor1 'S2 des-decode`$)$

{7} $(\text{rep}_d$ `Agent2 tractor2 'S3 direct-perc`$)$

`Agent2` believes that these IOR's are equal and, in a generic sense, is using the individuating set $\{S2, S3\}$ to collectively represent both `tractor1` and `tractor2`. This was stated in expressions {2} and {3}. Since `tractor1` is distinct from `tractor2`, it follows from formula {4} that `Agent2` has a compression misconception.

If `Agent2`'s model of the situation were accurate, expressions {8} – {11} would describe his mental state.

{8} $(\text{rep}_d$ `Agent2 tractor1 'S2 des-decode`$)$

{9} $(\text{rep}_d$ `Agent2 tractor2 'S3 direct-perc`$)$

{10} $(\text{rep}_C$ `Agent2 tractor1` $\{S2\})$

{11} $(\text{rep}_C$ `Agent2 tractor2` $\{S3\})$

If `Agent1` is to recover from a RIF that is caused by `Agent2`'s compression misconception, then it would be natural for `Agent1` to try to get `Agent2` to revise his mental state to match this description.

4.4 Representing Agent1's Initial Model of Agent2

We now write down a representation for `Agent1`'s model of `Agent2`'s mental state at the end of the fifth event, but before `Agent1` has correctly identified and diagnosed the existential misconception. Expression {12} states that `Agent1` models `Agent2` as having two distinct IOR's, one perceptual and one discourse-based, which represent `tractor1`. `Agent1` models these IOR's are being contained in an individuating set for `Agent2`. The individuating set $\{S1\}$ is part of `Agent1`'s vocabulary of term-expressions which `Agent1` uses to represent `tractor1`; the symbol "`tractor1`" is not part of `Agent1`'s vocabulary.

The rationale for constructing this representation is as follows. First, `Agent1` believes that `Agent2` has a representation for the object that `Agent1` told `Agent2` about. Second, `Agent1` believes that `Agent2` saw this

object when Agent2 entered the barn. Therefore, Agent2 should have generated an IOR for each of these information-acquisition processes. Furthermore, Agent1 believes that Agent2 should have recognized that these IOR's represent the same object and should therefore be included in the same individuating set.

```
{12} (bel Agent1
      '(∃e (∃s1 (∃s2
           (repd Agent2 {S1} s1 des-decode) ∧
           (repd Agent2 {S1} s2 direct-perc) ∧
           s1 ≠ s2 ∧
           s1 ∈ e ∧
           s2 ∈ e ∧
           (repc Agent2 {S1} e)))))
```

Agent1's model of Agent2 is inaccurate in two ways. First, Agent1, not knowing about tractor2, does not model Agent2 as having a representation for tractor2. Second, Agent1 does not model Agent2 as having a compression misconception. We have seen that Agent2 uses one individuating set (namely, $\{S2, S3\}$) to represent two distinct entities (namely, tractor1 and tractor2), but Agent1 models Agent2 as using one individuating set, e, to represent one entity as a result of two different information acquisition processes.

4.5 Representing Agent1's Model of Agent2 after Diagnosis

After Agent1 successfully diagnoses the existential misconception as the cause of the apparent belief discrepancy, Agent1's model will more accurately describe Agent2's actual mental state. In particular, expression {13} describes what Agent1's more accurate model of Agent2's mental state would look like.

```
{13} (bel Agent1
       '(∃e1 ∃s1 ∃s2 ∃o
            (repd Agent2 {S1} s1 des-decode) ∧
            (repd Agent2 o s2 direct-perc) ∧
            s1 ≠ s2 ∧
            s1 ∈ e1 ∧
            s2 ∈ e1 ∧
            o ≠ {S1} ∧
            (repc Agent2 {S1} e1) ∧
            (repc Agent2 o e1) ∧
            ¬∃e2 (repc Agent1 o e2)))
```

Expression {13} partially represents the situation described by node 3 in Figure 2. It also represents the situation depicted in Figure 3. In particular, Agent1 models Agent2 as:

1. representing tractor1 as a result of description decoding.
2. representing some object o as a result of direct perception.
3. representing that o is different than tractor1.
4. using one individuating set to represent two distinct entities (i.e., having a compression misconception).

Furthermore, Agent1 models itself as not having a representation for the object o.

4.6 Representing Agent1's Model of Agent2 after Recovery

Once Agent1 has diagnosed the RIF as resulting from divided reference stemming from a compression misconception, Agent1 can construct a representation of the unknown object. Let us assume that Agent1's IOR for this object is the symbol "S4" and that Agent1's individuating set for this object is $\{S4\}$. For our purposes, recovery from the RIF is achieved when Agent1 provides Agent2 with information sufficient to remove the misconception. Agent1's model of Agent2 would then be described by expression {14}.

```
{14} (bel Agent1
       '(∃e1 ∃e2 ∃s1 ∃s2
            (repd Agent2 {S1} s1 des-decode) ∧
            (repd Agent2 {S4} s2 direct-perc) ∧
            s1 ≠ s2 ∧
            e1 ≠ e2 ∧
            s1 ∈ e1 ∧
            s2 ∈ e2 ∧
            e1 ⋂ e2 = ∅ ∧
            (repc Agent2 {S1} e1) ∧
            (repc Agent2 {S4} e2)))
```

5 Inferential Requirements

With the representations in the previous section, the inferential requirements for the diagnosis and recovery problem for divided reference can now be more precisely stated. First, a note on terminology: when we use a phrase like "representation {12}" we mean the representation inside of Agent1's knowledge base that is described by expression {12}.

There are three parts to the specification. First, specify an inference process, along with required background knowledge, to allow Agent1 to initially construct representation {12}. Second, specify an inference process, along with required background knowledge to allow Agent1 to replace

representation {12}, with representation {13}. Third, specify a set of communication actions which Agent1 can take in order to change Agent2's mental state of Agent2 so that the mental state is correctly described by representation {14}.

5.1 Constructing the Initial Model

This section sketches how Agent1 might initially construct representation {12}. In the divided reference scenario, the agents generated IOR's as a function of their observations and the messages they received — that is, by processes of information acquisition. For example, during the first event Agent1 acquired beliefs

about a tractor as a result of an information acquisition process applied to observing the tractor. Similarly, during the second event, **Agent2** acquired beliefs about a tractor as a result of an information acquisition process applied to being informed of the tractor's existence. These are instances of the *description decoding processes* and the *direct perception processes* first mentioned in section 4.1.

Expressions {15}, {16}, and {17} below isolate the important subparts of expression {12}.

{15} (rep$_d$ **Agent2** {S1} s_1 **des-decode**)

{16} (rep$_d$ **Agent2** {S1} s_2 **direct-perc**)

{17} (rep$_c$ **Agent2** {S1} e)

Devising an inference process to allow **Agent1** to autonomously construct the equivalent of {15} and {16} is straightforward. s_1 and s_2 stand for IOR's that have been created directly as a result of processes of information acquisition. Therefore, **Agent1** can model the construction of these IOR's within **Agent2** if **Agent1** has two kinds of information. First, **Agent1** needs a model of each of the information acquisition processes that control the construction of these IOR's — one for description decoding and one for direct perception. Second, **Agent1** needs to know the inputs to these processes. Let us call these *information-acquisition input events*. With respect to determining input events, **Agent1** has the following information:

1. There is a tractor in the barn.
2. **Agent1** tells **Agent2** that *there is a tractor in the barn*.
3. **Agent2** enters the barn.

The information in the second expression allows **Agent1** to determine two things: 1) there was an instance of a description decoding process for **Agent2**; 2) the input event to the description decoding process was the statement *there is a tractor in the barn*. This information is sufficient to enable **Agent1** to construct expression {15}. Namely, **Agent1** uses its simulation of **Agent2**'s description decoding process as applied to this input event in order to generate the representation. In a similar fashion, the combined information in the first and third expressions enables **Agent1** to determine that there was an instance of a direct perception process for **Agent2** and that the input to this process was the observation of a tractor. An implementation of a program to construct statements like {16} by modeling an agent's perceptual processes appears in [21, p. 347].

One way for **Agent1** to construct a representation equivalent to expression {17} is for **Agent1** to simulate **Agent2**'s recognition process. This is because the act of **Agent2** concluding that s_1 and s_2 represent the same object, and therefore belong to the same individuating set, is a kind of recognition process for **Agent2**.

Expression {12} was constructed because **Agent1**'s information about inputs to **Agent2**'s information acquisition processes are inaccurate. In particular, **Agent1** concludes, perhaps by default, that the perceptual input for **Agent2** in the third event of the scenario is **tractor1**, but it is actually **tractor2**.

5.2 Constructing the Other Representations

In section 5.1, **Agent1** was equipped with three things:

1. The ability to simulate **Agent2**'s processes of information acquisition — both description decoding and direct perception.
2. The ability to simulate **Agent2**'s recognition process.
3. Knowledge of the input events to these processes.

This apparatus allowed **Agent1** to autonomously construct representation {12}. This apparatus can also be used to construct representations {13} and {14}.

The construction method is based on two ideas. First, it is easier to delete a flawed representation and build a new representation from scratch, rather than to revise an existing flawed representation. Second, the new representations are constructed the same way that the initial representation was constructed. Namely, input events are selected to be used as inputs to the simulations of the information acquisition and recognition processes. It follows from these strategies that **Agent1** will not try to modify representation {12} to construct the representation {13}. Instead, **Agent1** will try to construct representation {13} from scratch.

5.2.1 Constructing the Model after Diagnosis

The detection and diagnosis problems were initially described in sections 2.2.1 and 2.2.3. We stated that one of the KR requirements to allow **Agent1** to diagnose the cause of the apparent belief discrepancy that appeared at the end of the divided reference scenario is that **Agent1** be able to represent that **Agent2** has a compression misconception. This representation is given in expression {13}. This section discusses how **Agent1** might replace representation {12} by representation {13} after learning of the belief discrepancy.

Agent1 needs to infer that **Agent2** has observed a tractor other than **tractor1**, and that **Agent2** subsequently made the faulty recognition decision of concluding that this other tractor is the same as **tractor1**. We cannot explain how **Agent1** can make this inference, but given that **Agent1** has made the inference, we can explain how **Agent1** can autonomously construct representation {12}. Below we list the important subparts of representation {13}.

{18} (rep$_d$ **Agent2** {S1} s_1 **des-decode**)

{19} (rep$_d$ **Agent2** o s_2 **direct-perc**)

{20} (repc Agent2 {S1} e_1)

{21} (repc Agent2 o e_1)

{22} $\neg\exists e_2$ (repc Agent1 o e_2)

Let $\{S1\}$ be the individuating set that Agent1 uses to represent tractor1 and o be the existentially quantified variable that Agent1 uses to stand for the unknown tractor. Agent1 can construct expression {18} in exactly the same way as expression {15} was constructed. Expression {19} is constructed in the same manner as expression {16}, except that the input event for the simulation of the direct perception process is Agent2's observation of the unknown tractor. Unfortunately, the unknown tractor2 is sufficiently similar to tractor1 to cause Agent2 to make a faulty recognition decision. Agent1 can construct expressions {20} and {21} by simulating Agent2's recognition process.

The inference requirements to derive expression {22} are different than those for the previous expressions. In order for Agent1 to conclude that it does not have a representation for object o, Agent1 must infer that o is different from every object that it does have a representation for.

5.2.2 Constructing the Model for Recovery

After diagnosis, Agent1 now knows the actual situation in the tractor domain. In particular, there is a tractor which Agent1 does not have an explicit representation for. This provides enough information to allow Agent1 to construct a representation of that tractor. Let us arbitrarily call this representation $\{S4\}$. Agent1 can now construct representation {14} in three steps. First, Agent1 simulates Agent2's description decoding process when given the input event of being told about tractor1. Second, Agent1 simulates Agent2's direct perception process when given the input event of observing tractor2. Third, Agent1 simulates Agent2's recognition process where a faulty recognition decision is not made.

Agent1 now has the information to correct Agent2's compression misconception. First, Agent1 tells Agent2 to erase its previous representation. Next Agent1 supplies Agent2 with the appropriate input events to its information acquisition processes along with the warning to avoid a faulty recognition decision.

6 Conclusions

This divided reference problem applies to any agents that commmunicate using descriptions in domains containing newly discovered objects and is not specific to human communication in natural language. In such a domain, standard names cannot be used. To my knowledge, there was previously only one proposal for a computational model of reference that does not depend on standard names. This is the Appelt/Kronfeld proposal [2]. Unfortunately, their reference schema is undefined for situations of divided reference. The current paper develops the theory of IOR's and individuating sets, maps out representations for one of the communicating agent's mental state at crucial points in the divided reference scenario, states inference requirements for an agent to be able to operate in this kind of task domain, and sketches methods which may be used to meet these requirements.

6.1 Limitations of the Current Work

The representations used in section 4 assumed that Agent1 modeled Agent2 as explicitly using mental structures like individuating sets and IOR's. This amounts to attributing to Agent1 knowledge of a theory of commonsense representation. I believe this is a reasonable design decision for building autonomous agents that can communicate with other autonomous agents, but it should be acknowledged that previous formalisms have been rightfully criticized for doing this unwittingly [5]. If one were modeling human belief ascription, the methods of [4,6,3], which are based on folk-psychological metaphors, or the analogical mechanisms of [20,22], might be more appropriate. Unfortunately, since so little is known about metaphor-based inference, a metaphor-based or analogy-based belief model cannot, at the present time, be used as the basis of a technology for constructing autonomous agents that engage in description-based communication.

Acknowledgements

The ideas in this paper benefited from discussions with many people in various contexts. These include: Eugene Charniak, Steve Giambroni, Robert Goldman, R. Loganantharaj, Stu Shapiro, Padmini Srinivasan, Somnuk Keretho, Steve LaFleur, and Ay-Hwa Andy Liou. Thomas Ioerger critiqued an earlier draft of the manuscript.

References

[1] Douglas E. Appelt. Planning English referring expressions. *Artificial Intelligence*, 26(1):1–33, 1985.

[2] Douglas E. Appelt and Amichai Kronfeld. A computational model of referring. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 640–647, Milan, Italy, August 23–28, 1987.

[3] Afzal Ballim, Yorick Wilks, and John Barnden. Belief ascription, metaphor, and intensional identification. *Cognitive Science*, 15(1):133–171, 1991.

[4] John A. Barnden. Belief, metaphorically speaking. In *Proceedings of the First International Conference on Principles of Knowledge Representation*, pages 21–32, Toronto, Ontario, Canada, May 15–18 1989.

[5] John A. Barnden. Imputations and explications: Representational problems in treatments of propositional attitudes. *Cognitive Science*, 10(3):319–364, 1986.

[6] John A. Barnden. *Naive Metaphysics: A Metaphor Based Approach to Propositional Attitude Representation (Unabridged Version)*. Technical Report MCCS-90-174, New Mexico State University, 1990. Computing Research Laboratory, Box 30001, Las Cruces, New Mexico, 88003.

[7] Sehyeong Cho and Anthony S. Maida. *Reference Identification using a Bayesian Framework*. Technical Report CS–91–19, Department of Computer Science, Penn State University, University Park, PA 16802, 1991.

[8] Herbert H. Clark and Deanna Wilkes-Gibbs. Referring as a collaborative process. In Phillip R. Cohen, Jerry Morgan, and Martha E. Pollack, editors, *Intentions in Communication*, pages 463–493, MIT Press, Cambridge, MA, 1990.

[9] Phillip R. Cohen. The pragmatics of refering and the modality of communication. *Computational Linguistics*, 10(2):97–146, 1984.

[10] Robert Dale and Nicholas Haddock. Content determination in the generation of referring expressions. *Computational Intelligence*, 7(4), 1991.

[11] Ernest Davis. *Representations of Commonsense Knowledge*. Morgan Kaufmann Publishers, San Mateo, CA, 1990.

[12] Michael Genesereth and Nils Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, Los Altos, CA, 1987.

[13] Bradley A. Goodman. Reference identification and reference identification failures. *Computational Linguistics*, 12(4):273–305, 1986.

[14] Adam J. Grove and Joseph Y. Halpern. Naming and identity in a multi-agent epistemic logic. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference (KR'91)*, pages 301–312, Morgan Kaufmann, 1991.

[15] Andrew R. Haas. A syntactic theory of belief and action. *Artificial Intelligence*, 28(3):245–292, 1986.

[16] Graeme Hirst. Existence assumptions in knowledge representation. *Artificial Intelligence*, 49(1–3):199–242, 1991.

[17] David Kaplan. Quantifying in. *Synthese*, 19:178–214, 1968-69.

[18] Saul Kripke. A puzzle about belief. In Avishai Margalit, editor, *Meaning and Use*, D. Reidel, Boston, USA, 1979. Papers Presented at the Second Jerusalem Philosophical Encounter, April, 1976.

[19] Amichai Kronfeld. *Reference and Computation: An essay in applied philosophy of language*. Cambridge University Press, New York, 1990.

[20] Anthony S. Maida. Introspection and reasoning about the beliefs of other agents. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 187–195, Amherst, MA, August 15–17 1986.

[21] Anthony S. Maida. Maintaining mental models of agents who have existential misconceptions. *Artificial Intelligence*, 50(3):331–383, 1991.

[22] Anthony S. Maida, Jacques Wainer, and Sehyeong Cho. A syntactic approach to introspection and reasoning about the beliefs of other agents. *Fundamenta Informaticae*, 15(3,4):333–356, 1991.

[23] John McCarthy. Elephant 2000: A programming language based on speech acts. 1990. Unpublished manuscript.

[24] Nils J. Nilsson. Logic and artificial intelligence. *Artificial Intelligence*, 47(1–3):31–56, 1991.

[25] Donald Perlis. Languages with self-reference I: Foundations (or We can have everything in first-order logic!). *Artificial Intelligence*, 25(3):301–322, 1985.

[26] Yoav Shoham. *Agent-Oriented Programming*. Technical Report STAN–CS–1335–90, Computer Science Department, Stanford University, 1990.

IV.

Taxonomic Logics

"Reducing" CLASSIC to Practice: Knowledge Representation Theory Meets Reality

Ronald J. Brachman
AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, NJ 07974-0636

Abstract

Most recent key developments in research on knowledge representation (KR) have been of the more theoretical sort, involving worst-case complexity results, solutions to technical challenge problems, etc. While some of this work has influenced practice in Artificial Intelligence, it is rarely—if ever—made clear what is compromised when the transition is made from relatively abstract theory to the real world. CLASSIC is a state-of-the-art description logic with an ancestry of extensive theoretical work (tracing back over a decade to KL-ONE), and several novel contributions to KR theory. Basic research on CLASSIC paved the way for an implementation that is now used in practice, including by users not versed in KR theory. In moving from a pure logic to a practical tool, many compromises and changes of perspective were necessary. For the first time, we report on this transition and articulate some of the profound influences practice can have on relatively idealistic theoretical work. We have found that CLASSIC is quite useful in practice, and strongly retains most of its original spirit, but much of our thinking and many details had to change along the way.

1 INTRODUCTION

In recent years, the research area of knowledge representation (KR) has come to emphasize and encourage what are generally considered more "theoretical" results, such as novel logics, formal complexity analyses, and solutions to highly technical challenge problems. In many respects, this is reasonable, since such results tend to be clean, presentable in small packages, easily built upon by others, and directly evaluable because of their formal presentation. But lurking in the virtual hegemony of theory and formalism in published work is the tacit belief that there is nothing interesting enough in the reduction of KR theory to practice to warrant publication or discussion. In other words, it seems generally assumed that theoretical contributions can be reduced to practice in a straightforward way, and that once the initial theoretical work is wrapped up, all of the novel and important work is done.

Experience in building serious KR systems challenges these assumptions. There is a long and difficult road to travel from the pristine clarity of a novel rule of inference or a new logic to a system that really works and is usable by those other than its inventors. Events along this road can fundamentally alter the shape of a knowledge representation system, and can instigate substantial amounts of new theoretical work. My goal here is to attempt to show some of the key influences that KR practice can exert on KR theory. In doing so, I hope to reveal some important contributions to KR research to be made by those most concerned with the reduction to practice. In order to do this, I will lean on our experience with CLASSIC, a modern description logic. CLASSIC has seen quite concretely both ends of the theory-to-practice spectrum: on the one hand, it was developed after many years of KL-ONE-inspired research on description systems, was initially presented as a clean and simple logic, has a formal semantics, etc.; on the other, it has also been re-engineered from an initial LISP implementation to C, and is used in a fielded industrial product on an everyday basis (making CLASSIC one of at most a few KR systems to be moved all the way into successful practice). Our substantial system development effort has made it clear to us that moving a KR idea into real practice is not just a small matter of programming, and that significant research is still necessary even after the basic theory is in place.

After first giving a brief introduction to the goals and design of our system, I will summarize the "theoretical" CLASSIC as originally proposed (and published). I will then briefly explain our transition from research design to production use. Rather than give a potentially unrevealing historical account, I will subse-

quently attempt to abstract out five general factors that caused important changes in our thinking or in the system design, giving concrete examples of each in the evolution of CLASSIC:

1. general considerations in creating and supporting a running system;
2. implementation issues, ranging from efficiency tradeoffs to sheer complexity of building a feature;
3. general issues of real use by real people, such as learnability of the language and error-handling;
4. needs of particular applications; and
5. the unearthing of incompleteness and mistakes because of the unforgiving nature of programming.

While some of these concerns may seem rather prosaic, the point is that they have a hitherto unacknowledged—and substantial—influence on the ultimate shape, and certainly on the ultimate true value of any knowledge representation proposal. There is nothing particularly atypical about the research history of CLASSIC; as a result, the lessons to be learned here should apply quite generally in KR research.

2 THE ORIGINAL CLASSIC

While potential applications were generally kept in mind, CLASSIC was originally designed in a typical research fashion—on paper, with careful attention paid to formal, logical properties, and without much regard to implementation or application details. This section outlines briefly the original design, before we attempted to build a practical tool based on the formal logic (for more details on CLASSIC, *per se*, see [2, 4]).

2.1 GOALS

CLASSIC was designed to clarify and overcome some limitations in a series of knowledge representation languages that stemmed originally from work on KL-ONE [3, 6]. Previous work on NIKL [10], KRYPTON [5], and KANDOR [12] had shown that languages emphasizing structured descriptions and their relationships were of both theoretical and practical interest. Work on KL-ONE and its successors grew to be quite popular in the US and Europe in the 1980's, largely because of the semantic cleanliness of these languages, the appeal of object-centered (frame) representations, and their provision for some key forms of inference not available in other formalisms (e.g., *classification*—see below). Numerous publications in recent years have addressed formal and theoretical issues in "KL-ONE-like" languages, including formal semantics and computational complexity of variant languages (see, for example, [9]). However, the key prior implemented systems all had some fundamental flaws (e.g., NIKL had no assertion mechanism; KRYPTON was so general as to be unusable; KANDOR handled individuals in a very incomplete way), and the CLASSIC effort was in large part launched to design a formalism that was free of these defects.

Another central goal of CLASSIC was to produce a compact logic and ultimately, a small, manageable, and efficient system. Small systems have important advantages in a practical setting, such as portability, maintainability, and comprehensibility. Our intention was eventually to put KR technology in the hands of regular technical (non-AI) employees within AT&T, to allow them to build their own domain models and maintain them. Success on this account very strongly depends on how simply new technology integrates into an existing environment and how easy it is to learn how to use it, not to mention reasonable and predictable performance. Our plan was to take computational tractability seriously, but not to the point of dogmatism. If a proposed construct caused a key operation to become undecidable or intractable, we would first determine how critical it was to our users, and remove it if it were not important. If it were too critical to give up, we might propose an incomplete algorithm, but we would have to be very careful about any such proposal (we learned much more about what this meant as time went on, as we shall see). All in all, because of our desire to connect with extremely busy developers working on real problems, simplicity and performance were paramount in our thinking from the very beginning.

Finally, CLASSIC was designed to fill a small number of specific application needs. We had had experience with a form of deductive information retrieval, most recently in the context of information about a large software system [8], and needed a better tool to support this work. We also had envisioned CLASSIC as a deductive, object-oriented database system (see [2]; some success on this front was eventually reported in [14] and [7]). It was *not* our intention to provide some generic "general knowledge representation system," applicable to any and all problems. CLASSIC would probably not be useful, for example, for arbitrary semantic representation in a natural language system, nor was it intended as a "shell" for building an entire expert system. But the potential gains from keeping the system small and simple justified the inability to meet arbitrary (and too often, ill-defined) AI needs—and it was most important to have it work well on our target applications.

2.2 THE DESCRIPTION LANGUAGE

CLASSIC is based on a *description logic*, a formal logic whose principal objects are structured terms used to describe individual objects in a domain.[1] Descriptions

[1] We use the term *description logic* to better capture the essence of the approach than terms like "terminological logic," which have had some prior use. These logics do

are constructed by composing operators from a small set and specifying arguments to these operators. For example, using the operator **ALL** (a way to restrict the values of a property to members of a single class), we might construct the description, "(something) all of whose children are female":

```
(ALL child FEMALE).
```

Descriptions can then be asserted to hold of individuals in the domain, associated with names in a symbol table, or used in simple rules. Because of the formal, compositional structure of descriptions, certain inferences follow both generically—one description can be proven to imply another—and with respect to individuals—a description can imply certain non-obvious properties of an individual.

In CLASSIC, we call our descriptions *concepts*, and individual objects in the domain are modeled by *individuals*. To build concepts, we generally use descriptions of properties and parts, which we call *roles* (e.g., child, above). CLASSIC also allows the association of simple *rules* with named concepts; these are considered assertional, forward implications—in other words, simple universal statements about members of the class designated by the concept. For example,

```
AT&T-EMPLOYEE
⇒
(AND (AT-LEAST 1 payroll-acct-no)
     (ALL payroll-acct-no 6-DIGIT-INTEGER))
```

would mean "An AT&T employee has at least one payroll account number, all of which are 6-digit integers."

CLASSIC's description-forming operators are based on the key constructs seen in frame representations over the years. These operators have cleaned up ambiguities in prior frame systems, and are embedded in a fully compositional, uniform description language. The operators in the original design ranged from conjunction of descriptions (**AND**); to role "value restrictions" (**ALL**); number restrictions on roles (**AT-LEAST**, **AT-MOST**); a set-forming operator (**ONE-OF**); and operators for forming "primitive" concepts (**PRIMITIVE**, **DISJOINT-PRIMITIVE**), which have necessary but not sufficient conditions. CLASSIC also has a role-filling operator (**FILLS**) and an operator for "closing" roles (**CLOSE**),[2] although these were originally only applicable to individuals.

much more than just form terms. In addition, the broader term encompasses other related, but historically different work, such as OMEGA [1], which has much in common with the KL-ONE-based languages.

[2]Role fillers for CLASSIC's individuals are treated under a "non-closed-world" assumption, in that unless the KB is told that it knows all the fillers of a certain role, it assumes that more can be added (unless the number restrictions forbid it, in which case role closure is implied).

2.3 OPERATIONS ON CLASSIC KNOWLEDGE BASES

The description language allows the formation of the basic expressions that are used to define named concepts, create rules, and describe individual objects in the domain. In order to create new concepts and assign descriptions to individuals, etc., the user interacts with the CLASSIC system by means of a set of knowledge base-manipulating functions.

Generally speaking, operations on a CLASSIC knowledge base (KB) include additions to the KB and queries.[3]

Additive (monotonic) updates to the KB include the definition of new concepts or roles,[4] specification of rules, and assertion of properties to hold of particular individuals. One of the principal contributions of CLASSIC is its ability to specify incomplete information on individuals; for example, assertions like

```
Leland ∈ (AND (AT-LEAST 1 child)
              (ALL works-for
                   (ONE-OF Great-Northern
                           Double-R-Diner)))
```

are possible ("Leland has at least one child and works for either the Great Northern or The Double-R Diner"). Thus, individual objects are not required to be in the restricted form of simple tuples or complete records.

CLASSIC can also answer numerous questions about a KB, including whether one concept subsumes another, whether an individual is an instance of a concept, whether two concepts are disjoint, and various retrieval types of queries (e.g., fetch the properties of an individual, fetch all the instances of a concept).

These general operations on CLASSIC knowledge bases were characteristic of the system throughout its evolution, although specific operations changed in interesting ways, as discussed later.

2.4 INFERENCES

A key advantage of CLASSIC over many other systems is the variety and types of inferences it provides. Some of the standard frame inferences, like (strict) inheritance, are part of CLASSIC's definition, but so are several others that make description-logic-based systems unique

[3]As we shall see later, one of the results of the evolution being discussed here is our extension of these operations to include *retraction*, although in the original version of CLASSIC, nothing was said about removing or changing information.

[4]By "definition" here, we mean the association of a CLASSIC description with an atom in a symbol table, e.g., SATISFIED-GRANDPARENT ≐ (AND PERSON (ALL grandchild MARRIED)).

among object-centered KR systems. Among CLASSIC's inferences are

- *"completion" inferences*: logical consequences of assertions about individuals and descriptions of concepts are computed; there are a number of these inferences CLASSIC can make, including *inheritance*, *combination* of restrictions on concepts and individuals, and *propagation* of consequences from one individual to another;
- *contradiction detection*: inherited and propagated information are used to detect contradictions in descriptions of individuals, as well as incoherent concepts.
- *classification and subsumption inferences*: *concept classification*, in which all concepts more general than a concept and all concepts more specific than a concept are found; *individual classification*, in which all concepts that an individual satisfies are determined; and *subsumption*, i.e., whether or not one concept is more general than another.
- *rule application*: when an individual is determined to satisfy the antecedent of a rule, it is asserted to satisfy the consequent as well.

2.5 OTHER THEORETICAL ASPECTS OF CLASSIC

Since the original design of CLASSIC proceeded mainly from a theoretical standpoint, other formal aspects of the logic were explored and developed. CLASSIC had a traditional formal semantics (details are not critical here), similar to the semantics of related languages. While this was mostly a matter of formal "hygiene," we felt that having an intuitive and natural formal account of what things meant would be of help to users.

Given our desire for compactness and performance, we were also concerned with the computational complexity of the inferences to be provided by CLASSIC. While no formal proofs were published, we determined that subsumption—the key inference—was at worst polynomial, when applied to expanded definitions (and when **SAME-AS** was applied to functional roles only; see Section 4.2.1).[5] It also seemed clear at the time that a complete inference system could be implemented for our simple language, especially since this kind of system had been built several times before. Indeed, by the time the logic was first published, a prototype implementation was almost finished.

[5]Nebel later showed that description logics with expandable definitions were intractable [11]. But this result had little practical consequence for other languages in the KL-ONE family, just as similar results have had no real consequences for programming languages like Standard ML. Concept hierarchies never got deep enough to allow the term-expansion exponential to take over.

2.6 ANTICIPATING IMPLEMENTATION

The final form of the orginal CLASSIC proposal to some extent anticipated building a practical tool. Two concessions were made directly in the language. First, we allowed the formation of concepts that could compute their membership using *test functions* to be written in the host programming language. These would act as primitive concepts, in that their necessary and sufficient conditions would not be visible to CLASSIC. But they would allow the user to make up for CLASSIC's limited expressive power in some cases. Second, we specified a class of individuals in the language called *"host" individuals*, which would allow direct incorporation of things like strings and numbers from the host programming language. Many previous KR languages had failed to make a clean distinction between numbers and strings borrowed directly from the implementation and objects totally within the representation system. CLASSIC cleared this up even in the initial formal design.

2.7 THE RESULT: "PURE" CLASSIC

Prior to the completion and release of an implementation and its use in applications, CLASSIC thus had the description language grammar illustrated in Figure 1—this is roughly the version of CLASSIC published in 1989 [2], and was the initial basis for discussions with our development colleagues (see below). Using descriptions formed from this grammar, a user could define new named concepts and roles, assert that certain restricted types of descriptions applied to individuals, and create rules. Numerous straightforward types of KB queries were also available.

Even at this point, the CLASSIC logic made a number of important contributions, including full integration of host language objects in the formal specification and semantics, a fully compositional description language, and the ability to assert partial information about individuals. In this respect, the work was worth publishing; but in retrospect it was naive of us to think that we could "just" build it and use it.

3 THE TRANSITION TO PRACTICE

Given the simplicity of the original design of CLASSIC, we held the traditional opinion that there was essentially no research left in implementing the system and having users use it in applications. In late 1988, we concluded a typical AI programming effort, building a CLASSIC prototype in COMMON LISP. As the research LISP version was nearing completion, we began to confer with colleagues in a development organization about the potential distribution of CLASSIC within the company. Despite the availability of

```
<concept-expression> ::=  THING | CLASSIC-THING | HOST-THING | <concept-name> |        % built-in names
                          (AND <concept-expression>+) |                                % conjunction
                          (ALL <role-name><concept-expression>) |                      % universal value restriction
                          (AT-LEAST <positive-integer><role-name>) |                   % minimum cardinality
                          (AT-MOST <non-negative-integer><role-name>) |                % maximum cardinality
                          (SAME-AS (<role-name>+)(<role-name>+)) |                     % role-filler equality
                          (TEST <function> [<realm>]) |                                % procedural test
                          (ONE-OF <individual-name>+) |                                % set of individuals
                          (PRIMITIVE <concept-expression> <index>) |                   % primitive concept
                          (DISJOINT-PRIMITIVE <concept-expression> <group-index> <index>)
<realm> ::=               HOST | CLASSIC
<individual-expression> ::= <concept-expression> | <individual-name> |
                          (FILLS <role-name> <individual-name>) |                      % role-filling
                          (CLOSE <role-name>) |                                        % role closure
                          (AND <individual-expression>+)                               % conjunction
<rule> ::=                <concept-name> ⇒ <concept-expression>
```

Figure 1: The Original CLASSIC Grammar (comments in italics).

a number of AI tools, an internal implementation of CLASSIC held many advantages: we could maintain it and extend it ourselves, in particular, tuning it to our users; we could assure that it integrated with existing, non-AI environments; and we could assure that the system had a well-understood, formal foundation (in contrast to virtually all commercially available AI tools). Thus we initiated a collaborative effort to create a truly practical version of CLASSIC, written in C. Our intention was to develop the system, maintain it, create a training course, and eventually find ways to make it usable even by novices. Meanwhile, as the C effort progressed, we began to develop our first applications using the LISP prototype.

The issues and insights reported in the next section arose over an extended period of time, in which we collaborated on the design of the version of CLASSIC to be released, and developed four substantial and different types of applications, two of which were more or less along the lines we expected, and two of which were not originally anticipated (a configuration application and a limited NL application). We developed interim versions of the CLASSIC system and received feedback from users all along the way. Thus the "transition research" reported here was stimulated both by the need to construct a usable system in general and by the demands of real users in real applications.

4 FEEDBACK FROM PRACTICE TO THEORY

The process of implementing, re-implementing in C, and having application builders use CLASSIC and provide us with feedback, resulted in significant changes. Some of these arose because of simple facts of life of providing real software to real people, some arose because it is impossible to anticipate many key needs before people start to use a system, and some arose because it is almost impossible to truly complete the formal analysis before you begin building a system.[6] Here we outline some of the factors that had direct influence on technical aspects of CLASSIC.

4.1 CREATING AND SUPPORTING A SYSTEM

Even if the setting is not a commercial one, the intent to create and release a system carries certain obligations. In the case of CLASSIC, our development collaborators made one thing very clear: do not release software of which you are unsure. In particular, it is important not to include features in an initial release that you might choose later to remove from the language. Once someone starts to use your system, it is almost unpardonable to release a future generation in which features that used to be supported no longer are (at least with respect to the central representation language constructs). In our case, this forced us, for example, to abandon an operator we were considering that would allow the expression of concepts like "at least one female child." In the general case, such an operator rendered inference intractable and we wanted (as best we could) to avoid known intractability. However, if our users demanded it, we would have been happy to provide a reasonable partial implementation. Unfortunately, we did not have a handle on an incomplete algorithm that we could guarantee would only improve its completeness with subsequent releases.[7]

[6] This is by no means to say that such prior formal analyses are not also critical to the success of a project. Here my intention is simply to focus on the unheralded role that implementation and use play in the success and ultimate shape of KR systems. In a subsequent paper, we hope to address in a more balanced fashion the mutually supporting roles played both by theoretical and practical concerns.

[7] It is equally critical that an incomplete algorithm have a simple and understandable description of what it com-

Upward compatibility dictated that even with incomplete algorithms, subsequent versions of the system would still make at least all inferences made in earlier versions. Thus, we were better off from this perspective in keeping the language simple, and left this construct out of the initial released version.

There were other influences on the evolution of CLASSIC because of this prosaic, but important type of constraint. For a while, we had contemplated a role inverse construct (see Section 4.2.2). While we had not designed such a construct into the original specification, it appeared to be potentially very useful in our applications. We worked out several solutions to the problem, including a fairly obvious and general one that allowed inverses to be used for any role at any time. However, the cost appeared to be very high, and it was not even clear that we could implement the most general approach. As a result, we were forced to abandon any attempt to implement role inverses in the first released version of CLASSIC. We were not totally confident that we could stick with any initial attempt through all subsequent releases, and we did not want to start with an awkward compromise that might be abandoned later.

Another consideration is harder to be technical about or to quantify, but was equally important to us. As the system began to be used, it became clear that we needed to be as sure as possible that our operators were the best way to carve up the terminological space. In description logics, there are many variant ways to achieve the same effect, some of which are more elegant and comprehensible than others. Since we intended to put our software in the hands of non-experts, getting the abstraction right was paramount. Otherwise, either the system would simply not be used, or a second release with better operators would fail to be upward compatible. In CLASSIC's case, we put a great deal of effort into an assessment of which operators had worked well and which had not in previous systems.

4.2 IMPLEMENTATION CONSIDERATIONS

There were at least two aspects of the implementation effort itself that ultimately ended up influencing the language and operation design. One was the sheer difficulty of implementing certain inferences, and the other was the normal kind of tradeoff one makes between time and space.

putes. Otherwise, users will invariably expect certain conclusions to be drawn when the algorithm in fact will miss them. Such mismatched expectations could be disastrous for the acceptability of the product. We were strongly told that it was better to eliminate a construct than to have a confusing and unintuitive partial implementation of it.

4.2.1 Complex operations

We began our implementation by attempting to construct a complex subsumption algorithm that included **SAME-AS** relations between roles. For example, we wanted to be able to detect that a concept that included

```
(AND (SAME-AS (boss) (tennis-partner))
     (SAME-AS (tennis-partner) (advisor)))
```

was subsumed by (i.e., was more specific than) a concept that included **(SAME-AS (boss) (advisor))**; in other words, that someone whose boss is her tennis-partner, and whose tennis-partner is her advisor, must of necessity be someone whose boss is her advisor.

Because **SAME-AS** could take arbitrary role *paths*, roles could possibly be unfilled for certain individuals, and roles could have more than one filler, this made for some very complex computations in support of subsumption. Cases had to be split, for example, into those where roles had some fillers and those where they had none, and less-than-obvious multiplications and divisions had to be made. More than once, as soon as we thought we had all cases covered, we would discover a new, more subtle one that was not covered. When we finally came to a point where we needed a result from group theory and the computation of a factorial to get some cardinalities right, we decided to abandon the implementation.

As it turns out, our attempts at a straightforward and efficient solution to what appeared to be a tractable problem were thwarted for a deep reason: full equality for roles is undecidable. This result was later proved by Schmidt-Schauss [13]. Thus, our implementation enterprise could never have fully succeeded. The end result of all of this was an important change to the CLASSIC language (and therefore needed to be reflected in the semantics): we moved to distinguish between *attributes*, which could have exactly one filler, and other roles, which could have more than one filler. **SAME-AS** could then be implemented efficiently for attributes, and the language was appropriately restricted to reflect the dichotomy. In the end, the distinction between attributes and multiply-filled roles was a reasonably natural one, given the distinction between functions and relations in first-order logic, and the common use of single-valued relations in feature logics and relational databases.[8]

Other aspects of CLASSIC evolved for similar reasons. For example, while our original specification for a **TEST** concept was conceptually sufficient, we left it to

[8] Attributes correspond to columns in relations, whereas multiply-filled roles would most easily correspond to two-column relations. These correspondences turned out to be of great use to us when we subsequently attempted to interface CLASSIC to a relational database [7].

the user to specify (optionally) a "realm" (i.e., *CLASSIC* or *HOST*). This made the parsing and structure normalization more complex—and different—for concepts that had **TESTs** than for those that did not. (It was also the only case of a concept construct for which the user had to specify a realm at all.) In order to make the code more simple and reliable (and the interface more uniform), we eventually substituted for **TEST** two different operators (**TEST-C**, **TEST-H**), which would each be unambiguous about its realm.[9]

4.2.2 Implementation tradeoffs

The compromises on space and time to be made when creating computer programs are well known. And in most cases, decisions made because of space or efficiency should not be of consequence to the system's functional interface. However, some tradeoffs can be extremely consequential, and yet never occur to designers of an unimplemented language.

One tradeoff that affected our user language involved the form and utility of role inverses. If one could afford to keep backpointers from every filler back to every role it fills, then one could have an (**INVERSE** <role>) construct appear any place in the language that a role can. This would be quite convenient for the user and would contribute to the uniformity of the langnage—but it would also entail significant overhead. An alternate view is to force users to explicitly declare in advance any role inverses that they intend to use at the same time the primitive roles are declared. Then, backpointers would be maintained only for explicitly declared roles. The point here is not which approach is best, but rather that practical considerations can have significant effects on a pure language that never takes such things into account. Given the large difference between the two approaches, and inferential difficulty that results from including inverses in the language, we chose to exclude them altogether from the first release. The latest version of CLASSIC (Version 2.0) now has them, since we have had a chance to think hard about the interplay between language, operation, and implementation design.

4.3 SERVING THE GENERAL USER POPULATION

Real use of a system, regardless of the particular applications supported, immediately makes rigorous demands on a formalism that otherwise looks good on paper. I consider two types of issues here: (1) can users comprehend and become facile with the logic you are using? and (2) critical features for usability of the system itself.

[9]In retrospect, we should have done this also for the **ONE-OF** construct, but have yet to do so.

4.3.1 Learnability/Usability of the Language

Concepts like "usability" are admittedly vague, but it is clear that users will not stick with a system if the abstractions behind its language and interface do not make sense. Formal semantics makes precise what things mean, and it behooves us to provide such formal bases for our logics. However, how simple a language is to learn and how easy it is to mentally generate the name of a function to be used are more likely the *real* dictators of ultimate success or failure.

In the CLASSIC world, this meant (among other things) that the language should be as uniform as possible—the more special cases, the more problems. Just being forced to think about this led us to an insight that made the language better: there was in general no good reason to distinguish between what one could say about an individual and what one could use as part of a concept. The **FILLS** constructor should have been equally applicable to both; in other words, it makes sense to form the general concept of "an automobile whose manufacturer is Volvo," where Volvo is an individual. In the original specification, we thought of role-filling as something one does exclusively in individuals. The one sticking point to a generalization was the **CLOSE** operator, which did not make much sense for concepts; but as we see below, the thinking about **CLOSE** that was instigated by user concerns eventually led us to determine that it was mistakenly in the language in the first place. As a result, the types of descriptions allowable as definitions of concepts and for assertions about individuals could be merged.

There were other simplifications based on generic user concerns like understandability that helped us derive an even cleaner logic. For example, the **PRIMITIVE** and **DISJOINT-PRIMITIVE** concept-forming operators, which we found problematic for non-experts in actual use, were removed from the language and better instantiated as variants on the concept-defining interface function. The conceptually adequate but awkward arguments to **DISJOINT-PRIMITIVE**s were also simplified; we decided to use global rather than local partitions of disjoint definitions.

Finally, we found that as we attempted to bring our logic to potential users, they urgently needed tutorials and sample knowledge bases—much more than they needed "bells and whistles" in the system. The more examples and guidance on "tricks of the trade," the better (although somewhat ironically, people at the same time seem to demand that less be necessary to read in order to get started). While it is common for us as researchers to spend most of our time extending the coverage of a logic, making an implementation faster, or building a graphical user interface, the most elegant formal system in the world is almost meaningless to users unless supporting infrastructure is available. In our case, we went so far as to write an article dis-

cussing typical knowledge bases, useful tricks, ideas in our logic that would be difficult for non-KR people, and a conventional methodology for building CLASSIC KB's [4]. We also collaborated on an internal training course, and developed sample knowledge bases for distribution with the system.

4.3.2 System Features

Even with a perfectly understandable and intuitive logic, a minimal, raw implementation will be almost impossible to use. In general, our customers told us, the system's development environment (for building and debugging knowledge bases) was a make-or-break concern. For example, logics discussed in papers do not deal with issues like *error-handling*, yet real users can not use systems unless they get meaningful error-reporting and reasonable error-handling, especially when the KR system is embedded in a larger system. As a result of direct and strong feedback from users, the released version of CLASSIC had extensive error-handling, including well-documented error codes and rational and consistent error returns.

One of the key aspects of error-handling that arose in our applications was the need to roll back the effects of an update that caused an inconsistency in the knowledge base. Since CLASSIC was able to do elaborate propagation and rule-application, a long and ramified inference chain may have been triggered before a problem was found, and unless every piece of that chain were removed, the knowledge base would be left in an unwanted state. This need led us to consider ways to unravel inferences, including a database-style "commit" operation (i.e., the knowledge base would never be changed until all inferences concluded successfully). We eventually settled on a more conventional AI approach using dependencies, which gave us a general facility that not only would guarantee the KB to be left in a meaningful state after an error occurred, but would allow the user direct *retraction* of facts previously told. This was the basis for an extremely useful "what if" facility in our configuration application. As it turned out, the availability of such a retraction capability was critical in "selling" the application to its sponsors.

Another generic area that needs attention is *explanation* of reasoning; this has generally been ignored by the KR community. If users are to build nontrivial KB's, they will need help in debugging them. They will need to know why an inference failed, or why a conclusion was reached. While the expert systems community may have learned this lesson, it is an important one for us in general KR as well. Our users made a very strong case to us that such a feature was critical to their successful construction of knowledge bases. While we learned this lesson in evolving the released version, there was no real explanation code in the first release. We have recently, however, made significant progress on a simple explanation facility for CLASSIC inferences, and this now appears in the most recent version of the system.

Another key point of tension between theory and practice is the notion of an *"escape"* for the user, e.g., a means to get around an expressive limitation in the logic by resorting to raw LISP or C code. As mentioned above, we reluctantly included in the original specification a **TEST** construct, which allowed the user to resort to code to express sufficiency conditions for a concept. In the original paper, we did not technically include **TEST**s in concept definitions, since no formal semantics was available for it. But our view was that this was not intended to be a general programming interface, and in the end we did develop guidelines for use (e.g., avoidance of side-effects) that could guarantee that **TEST**-defined concepts could fit into our formal semantics, even if the code itself is opaque to CLASSIC.

As it turned out, **TEST**-concepts were one of the absolute keys to successful use of CLASSIC. In fact, they not only turned out to be a critical feature to our users, but as we observed the patterns of tests that were written in real applications, we were able to ascertain a small number of new features that were missing from the language but fundamental to our applications. First, we discovered that users consistently used **TEST**s to encode simple numerical range restrictions; as we mention below, this led us to create **MAX** and **MIN** constructors for our concept language. Later, in one significantly large and real-world application, we found only six different patterns of **TEST** concepts, with over 85% of these falling into only two types; one was computing via a function a universal restriction on a role (actually, a numerical range), and the other was computing a numerical filler for a role (a simple sum). We are now making additions to CLASSIC to accommodate these common patterns of usage, and expect that the new version of the same knowledge base will be substantially simpler, and less prone to error (the original **TEST**s were written to achieve some of their effects by side-effect, which we can now eliminate).

Thus, while our original fear was that an escape to LISP or C was an embarrassing concession to implementation, and one that would destroy the semantics of the logic if used, our **TEST**s were never used for arbitrary, destructive computation. Rather, this mechanism turned out to be a means for us to measure specifically where our original design was falling short, all the while staying within a reasonable formal semantics.[10]

[10] As evidence of the continuing general lack of appreciation of the more theoretically inclined towards pragmatic issues, consider that one of the reviewers of our 1989 paper [2] called **TEST**s "an abomination." Yet, not only were they undeniably critical to our users, we managed to keep them in line semantically, and also learned from

4.4 MEETING THE NEEDS OF PARTICULAR APPLICATIONS

As soon as the system is put to any real use, mismatches or inadequacies in support of particular applications become very evident. In this respect, there seems to be all the difference in the world between the few small examples given in research papers and the details of real, sizable knowledge bases. As mentioned, we took on four significant and different types of application. While the demands from each of them were somewhat different, they clearly did *not* demand that we immediately extend CLASSIC to handle the expressive power of full first-order logic. In fact, the limited number of extensions and changes that arose from the interaction with individual applications are useful in all of them, and all stay within the original spirit of simplicity.

Among the first needs we had to address was the significance of numbers and strings. Virtually all of the applications needed to express concepts limiting the values of roles that had *HOST* values in them, as in, for example, "a manager whose salary is between 20000 and 30000." On the one hand, this need vindicated our original decision to integrate host information in a serious manner.[11] On the other, as mentioned above, the need to create TEST-concepts to test simple ranges like this showed us that we would have a hard time measuring up to almost any application that used real data (especially if it came from a DBMS). Thus, the most recent versions of CLASSIC have new concept types that represent ranges of *HOST* values.[12] These are integrated in a uniform manner with other concept constructors, and the semantics accounts for them.

Another major consequence of dealing with a significant application is the reality of *querying* the KB. Our original design of CLASSIC (as is the case with virtually all frame systems) paid scant attention to queries other than those of the obvious sort, e.g., retrieving instances of concepts. Once we began to see CLASSIC as a kind of deductive database manager, we were forced to face the problem that our querying facilities were very weak. This led to the design of a substantial query language for CLASSIC, which can handle the needed object-oriented queries, as well as the SQL-style retrievals that are so common in the real world of information management. While this is not profound (although the query language we developed has some novel features and is itself an important contribution), the key point is that it was the attempt at application that made us realize that an entire critical component was missing from our work.

Two other consequences of this sort bear brief mention.

First, our simple notion of a **TEST** was sufficient to get us off the ground. Our intention was to pass the individual being tested to the function as a single argument. As it turned out, our users needed to pass other items in as arguments. For example, if the test was a simple function to compute whether the value of a role was greater than some number, the number would have to be built directly into the test function; this in turn would lead to propagation of many almost-identical functions—unless we provided the ability to pass in additional arguments. We have done so in the latest versions of CLASSIC.

Second, our original design of rules was a sufficient foundation, but it required a named concept to exist as the left-hand-side of the rule. As soon as some of our users tried to use this, they found that they had to construct concepts artificially, just to serve to invoke the rules. While this posed no conceptual problem for the logic, and no semantic aberration, it became a practical nightmare. Thus, it was important to extend our rules to allow a filter; in other words, the rule could be associated with the most general named concept for which it made sense, but only fired when a filtering subcondition was satisfied. This now avoids needless creation of artificial concepts.

4.5 UNDERSTANDING WHAT LOOKED GOOD ON PAPER

Probably more commonly than researchers would like to admit, theoretical KR papers—especially for conferences—are not always what they seem. While theorems and formal semantics help us get a handle on the consequences of our formalisms, they do not always do a complete job (it is also not unheard of for them to contain mistakes). Our experience has been that several parts of our original formalism were clarified substantially by the experience of having to implement an inference algorithm and have it used on real problems. In each case, a change was necessary to the original formal work to accommodate the new findings.[13]

---

them some key things to do to make our system even more usable.

[11] We should point out that integration here is not just a simple matter of allowing numbers or strings in roles; it has ramifications for the language syntax and parsing, part of the concept hierarchy must be built automatically, data structures for *CLASSIC* individuals need to be carefully distinguished from arbitrary LISP structures, etc.

[12] **MAX** and **MIN** have instances that are numbers; e.g., **(MAX 25)** represents the set of integers that are less than or equal to 25. These are used to restrict the *value* of a filler of a role; for example, we could use **MAX** to specify the value restriction on a person's age, as in **(AND PERSON (ALL age (MAX 25)))**. **AT-LEAST** and **AT-MOST**, on the other hand, restrict the *number of fillers* of a role, not their values.

[13] In a sense, issues like those mentioned in this section are a result of less than complete formal analysis before

For example, we had originally proposed that **CLOSE** could appear in a description applied to an individual, to signal that the role fillers asserted by the description were the only fillers. Thus, one could assert of Dale,

```
(AND (FILLS friend Audrey)
     (FILLS friend Harry)
     (CLOSE friend));
```

this was to mean that Audrey and Harry were Dale's *only* friends. The semantics characterized this as a simple predicate closure operation. However, once a real knowledge base was constructed, and users started interacting with the system, we discovered a subtle ordering dependency: pairs of **CLOSE** operators could produce different effects if their order were reversed. This led us to discover that our original characterization of **CLOSE** was in general wrong. In reality, it had an autoepistemic aspect, and thus closing roles had to become an operation on an entire knowledge base, rather than a concept-forming constructor. It was thus removed from the description language and made a knowledge base operation.

We had a small number of similar experiences with other aspects of the language, for example, **TEST** constructs. Given when they would be invoked, it eventually became clear (thanks to a key user's discovery) that **TEST**s, which were originally two-valued functions, had to be *three*-valued—since CLASSIC supports partial information about individuals, it is possible for a test to "fail" at one point and succeed later, even with strictly monotonic additions to the KB. If the test truly failed the first time, and the individual were determined *not* to satisfy a description based on this failure, nonmonotonicity would be introduced in an inappropriate way. Thus functions now need to tell their caller if the individual provably satisfies the test, provably fails it, or currently fails but could possibly succeed later.

5 REAL CLASSIC

The result of the long and arduous trail implied above, from typical research paper to practical system, was a significant improvement in the CLASSIC language and the clarity of operations on a CLASSIC knowledge base. The grammar was simplified and made more uniform (see Figure 2), and the semantics was adjusted to be truer to our original intention. KB operations were streamlined and made more useful, and error-handling and retraction were added. The resulting system is unarguably superior to the original in every way: it has new constructs that meet real needs, substantial parts of it have been validated by use, the overall interface makes more sense, it is cleaner and more elegant, and it is devoid of flaws that were subtly hidden in the original.

The effects of the pragmatic factors I have described here are varied, and not easily classified. But they are clearly substantial and were critical to the success and ultimate form of CLASSIC. To summarize, here are some of the most important changes that were driven by the attempt to "reduce" the system to practice and put it to the test of real use:

- *language improvements:* distinction between attributes and multiply-filled roles; **SAME-AS** applicable to attributes only and efficiently computable; arguments for **TEST**-concepts; three-valued **TEST**s; completely compositional language with no order dependencies; equal descriptive power for individuals and concepts; numeric range concepts; rules have filter conditions, no longer requiring artificial concepts; realms of **TEST**-concepts unambiguous and **TEST** constructs made uniform with other parts of language;
- *interface improvements:* primitive and disjoint primitive definition as KB operators; disjoint primitive specification simplified; **CLOSE** as a KB operator; sophisticated query language and implemented query processor;
- *system features:* comprehensive error-reporting and handling; explanation facility; renaming of concepts.

6 LESSONS

Although a complete formal specification of a knowledge representation system (including an algorithmic specification of the inferences that the system is required to perform and a computational analysis of these inferences) is vitally important, the presence of a formal account is not sufficient for the success of the system. There is no guarantee, for example, that a formally tractable knowledge representation system can be effectively implementable, as it may be exceedingly difficult to code the inference algorithms or other portions of the system efficiently enough for use or perspicuously enough to tell if they are correct. Even then, there is no guarantee that the resulting system will be useful in practice, even if it appears at first glance to meet some apparent needs. Finally, getting the formal specification *really* right is an extremely difficult task, especially for systems that perform partial reasoning or which have non-standard but useful con-

implementation and use. The point is that because of the complexities and subtleties of real-world problems, and the extreme difficulty of anticipating in the abstract what real users will want, this type of effect is inevitable, and a critical contribution of practice over pure "theory." The history of KR in AI is fraught with the discovery of bugs and inadequacies in prior published formal papers, even though the originals were considered correct by both authors and reviewers.

```
<concept-expression> ::=  THING | CLASSIC-THING | HOST-THING | <concept-name> |
                          (AND <concept-expression>+) |
                          (ALL <role-name><concept-expression>) |
                          (AT-LEAST <positive-integer><role-name>) |
                          (AT-MOST <non-negative-integer><role-name>) |
                          (FILLS <role-name> <individual-name>+) |          % added for uniformity
                          (SAME-AS (<attribute-name>+) (<attribute-name>+)) |   % clarified
                          (TEST-C <function><arg>*) |          % clarified; arguments added; 3-valued
                          (TEST-H <function><arg>*) |          % clarified; arguments added; 3-valued
                          (ONE-OF <individual-name>+) |
                          (MAX <number>) |                                  % added
                          (MIN <number>)                                    % added
<individual-expression> ::= <concept-expression> | <individual-name>   % made uniform with concepts
<rule> ::=                (<concept-name><filter concept-expression>) ⇒ <concept-expression>
                                                                  % made more practical
```

Figure 2: The Resulting CLASSIC Concept Language (with comments).

structs. All told, *the implementation and use of the system is a vital complement to work on knowledge representation "theory."* It can illuminate problems in the formal specification, and will inevitably provide real problems for the theory side to explain.

Our experience with CLASSIC has taught us this lesson in some very specific ways. Any hope of having the system make a real impact (e.g., in a product) rested on some very practical considerations that in some cases were impossible to anticipate before interacting with developers. We learned through extensive interaction with our developers that issues like upward compatibility and simplicity were in some ways much more important than individual features. We learned that attention to complexity (although not maniacal concern with it) was very much worth the effort, because of the critical impact of performance and predictability on acceptance of the system. We also learned that we could not afford to be totally rigid on any point—be it language features, complexity, or names of functions—without jeopardizing potential use of the system. The feeling of the CLASSIC group is that the resulting system is clearly far better than anything we could have built in a research vacuum. And the effort of reducing our ideas to a practical system generated a great deal of research—on language constructs, complexity, and even formal semantics—that was not only interesting, but important simply by virtue of the very fact that it arose out of real problems.

At a more strategic level, one very important lesson for us was the significance of a certain kind of conservatism. We could have invested a large amount of time designing features and providing expressive power (and implementation complexity) that would have, as it turned out, gone completely to waste. On the flip side, our users gave us clear and direct evidence of features that they did need, and that we were not providing, via our **TEST** construct, which, to be honest, surprised us both in its criticality and in the simple and regular ways in which it was used—not to mention the smallness of the number of needed extensions. All told, our decision to start with a small (but not hopelessly impoverished) language, with room for growth in a reasoned fashion, was clearly a successful one. While such emphasis on simplicity might not necessarily be right for all projects, given the constraints under which product developers live, it is a key issue to consider when the practice that we are "reducing" to is not just for AI research but for development and product.

In sum, a number of key factors of a strongly pragmatic sort show that logics that look good on paper may have a long way to go before they can have any impact in the real world. These factors range from upward compatibility and system maintenance to implementation tradeoffs and critical system features like error-handling and explanation. They include learnability of the language and occasional escapes to circumvent limitations of the system. While individual gains in CLASSIC derived from attention to these practical concerns may each have been small, they all added up, and made a big difference to success of the system as a whole. It is quite clear that if practical concerns were ignored, the resulting system would have had at best limited utility. In fact, in general in our field, it seems that the true theoretical work is not done until the implementation runs and the users have had their say.

Acknowledgements

CLASSIC is the result of a substantial team effort. Those who brought the system to fruition include Alex Borgida, Deborah McGuinness, Peter Patel-Schneider, Lori Alperin Resnick, Gregg Vesonder, Elia Weixelbaum, and Jon Wright. Substantial contributions have also been made by Tom Kirk and Kevin Zalondek, and also Hector Levesque.

References

[1] G. Attardi and M. Simi. Consistency and completeness of OMEGA, a logic for knowledge representation. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 504–510, Vancouver, British Columbia, August 1981.

[2] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 59–67, June 1989.

[3] R. J. Brachman. *A Structural Paradigm for Representing Knowledge*. PhD thesis, Harvard University, Cambridge, MA, 1977. Revised version published as BBN Report No. 3605, Bolt Beranek and Newman, Inc., Cambridge, MA, July, 1978.

[4] R. J. Brachman, A. Borgida, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Resnick. Living with CLASSIC: When and how to use a KL-ONE-like language. In John Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, San Mateo, CA, 1991.

[5] R. J. Brachman, R. E. Fikes, and H. J. Levesque. KRYPTON: Integrating terminology and assertion. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 31–35, Washington, D. C., August 1983.

[6] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, April–June 1985.

[7] R. J. Brachman, P. G. Selfridge, L. G. Terveen, B. Altman, A. Borgida, F. Halper, T. Kirk, A. Lazar, D. L. McGuinness, and L. A. Resnick. Knowledge representation support for data archaeology. In *First International Conference on Information and Knowledge Management*, November 1992.

[8] P. Devanbu, R. J. Brachman, P. G. Selfridge, and B. W. Ballard. LaSSIE: A knowledge-based software information system. *Communications of the ACM*, 34(5):34–49, May 1991.

[9] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proceedings of the Twelfth International Conference on Artificial Intelligence*, pages 458–463, Sydney, Australia, August 1991.

[10] M. G. Moser. An overview of NIKL, the new implementation of KL-ONE. Technical Report 5421, BBN Laboratories, 1983. Report titled "Research in Knowledge Representation for Natural Language Understanding—Annual Report, 1 September 1982–31 August 1983".

[11] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2):235–249, May 1990.

[12] P. F. Patel-Schneider. Small can be beautiful in knowledge representation. In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, pages 11–16, Denver, December 1984. Revised and extended version available as AI Technical Report Number 37, Schlumberger Palo Alto Research, October 1984.

[13] M. Schmidt-Schauss. Subsumption in KL-ONE is undecidable. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proceedings of the First International Conferenceon Principles of Knowledge Representation and Reasoning*, pages 421–431. Morgan Kaufmann, May 1989.

[14] P. G. Selfridge. Knowledge representation support for a software information system. In *Proceedings of the Seventh IEEE Conference on AI Applications*, pages 134–140, Miami Beach, Florida, February 1991.

Towards the Systematic Development of Description Logic Reasoners: CLASP reconstructed

Alex Borgida
Dept. of Computer Science
Rutgers University
New Brunswick, NJ 08903

Abstract

It is argued that considerable benefits can be obtained by making KR&R systems easily extendable, so that new language constructs can be added on a per-application basis. In order to achieve this extensibility we need techniques for formally specifying the extensions, and for modularly and locally modifying the implementations. We present two such techniques applicable to the family of reasoners based on Description/Terminological Logics, namely natural semantics rules of inference, and the PROTODL customizable KBMS architecture. The bulk of the paper aims to demonstrate the efficacy of these techniques, together with some heuristics for their use, by showing how we could reconstruct a previously proposed description logic: Devanbu and Litman's extension to CLASSIC, called CLASP, designed to reason about actions and plans. In the process, we uncover a few deficiencies in the original proposal, and provide for the first time a formal semantics for CLASP.

1 Introduction and Aims

Knowledge representation and reasoning systems usually manage information about individuals, related by binary relationships (called roles and attributes — the latter are functional roles) and grouped into classes. The family of terminological or description logic reasoners[1] is distinguished by the fact that the classes in question are intensional entities with a compositional structure, which allows them to be involved in inferences. In fact, class definitions can be seen as variable-free terms, called *descriptions*, which are built up from identifiers using term/description constructors. For example, the description **and(PROCEDURE, all(params,ARRAY-VARS), at-least(3,params))** is composed, using the constructor **and**, from three subterms, and denotes individuals in the intersection of the respective denotations:

- **PROCEDURE**
  — the name of a description defined elsewhere;
- **at-least(3,params)**
  — individuals with at least three fillers for the **params** role;
- **all(params,ARRAY-VARS)**
  — individuals all of whose **params** fillers are instances of the description **ARRAY-VARS**.

Reasoning with descriptions is based on a logic built around the *subsumption* relationship, which corresponds to entailment between unary predicates.

There is convincing evidence that it would be useful to have the ability to *extend easily* KR&R systems with new kinds of inferences. In addition to earlier arguments for this stance (e.g., [Greiner & Lenat, 1980, Genesereth 1983, Lenat & Guha, 1990, Rowley *et al*, 1986, Gaines 1991]), we mention two particular benefits we would hope to achieve: It would facilitate the design and implementation of *domain-specific* DL-reasoners that are now being proposed in the literature, such as ones dealing with time (e.g., [Schmiedel, 1990]), planning (e.g., [Devanbu & Litman, 1991, Heinsohn 1992]), and constraint-networks (e.g., [Weida & Litman, 1992]). And it would provide an alternate approach to the the well-known expressiveness vs. computational cost standoff in KR&R system design [Doyle & Patil, 1991]: start with a limited language and extend it when its boundaries impinge; but tailor the extension to the specific problem being solved, so that in the case of highly intractable, or even undecidable language constructs, one can implement a subset of the valid inferences — those needed for the application at hand. Assuming that extensibility is a desirable property, how can we achieve it? In looking around for an analogy, we find that modern pro-

[1] See [Woods & Schmolze, 1992] for a review.

gramming language theory and compiler technology (PL&C) (e.g., [Aho *et al*, 1986]) supports language extensibility: for example, to add a case-statement to an Algol-like language we can first *specify* its syntax, type-correctness rule, and semantics; and we can then change the compiler *implementation* by modifying some of its modules (e.g., lexical/syntactic/semantic analyzers, intermediate code generator) but usually not others (e.g., symbol table, code generator). An important feature of the compiler modification is that *changes tend to be local and systematic*, such as adding a new clause to LEX or YACC parser generator, or adding a new case to the syntax-directed translation scheme. It is also significant that the changes correlate with the formal specifications. Contrast this with the work needed to change something like the first Fortran and its monolithic compiler, or, regretfully, most current KR&R systems.

The aim of this paper is to demonstrate that we are on the way to achieving a similar desirable state of affairs in KR&R, if we focus on the relatively homogeneous family of Description Logic Reasoners. Specifically, this paper aims to provide (further) evidence for the following claims: (1) The inferences that are performed by a DL-reasoners should be extended through the addition of new, possibly domain-specific, description constructors. (2) *Natural semantics axiomatizations*, as used in [Borgida 1992a], are a useful alternative way to present the semantics of DLs. (3) PROTODL [Borgida & Brachman, 1992] offers an architecture for *effective* DL reasoners that allows DL implementations to be built up from small components, which are either prefabricated or need to be filled in by the implementors. (4) The previous techniques, combined with heuristic guidelines for their use, can lead to a *more systematic* approach to specifying and implementing extensions to DLs, resulting in systems that are more likely to be correct and useful.

The above claims are empirical in nature: we argue for them by showing that the techniques mentioned can be used to achieve useful results. In the previously cited papers we have shown how natural semantics and PROTODL can be used to describe the CLASSIC system [Borgida *et al*, 1989]. In this paper, we propose to revisit CLASP – Devanbu and Litman's DL to support reasoning about actions and plans [Devanbu & Litman, 1991]. In some sense this is an independent test of the proposed approach, applied to a practically significant system: CLASP was used in a Software Information System [Devanbu *et al*, 1991]. By being able to reconstruct the actual implementation of CLASP, we provide evidence for claim (3), and by finding lacunae and alternatives not considered in the original paper on CLASP, we give additional credence to claims (1),(2), and (4).

Our plan is to introduce the techniques for specifying and implementing extensions using a simple and familiar example: the language construct **at-most**, which bounds the number of objects that can be related by relationships. We then revisit CLASP's methods for representing states and actions, and after uncovering some difficulties with actions, we specify and implement a new proposal. Next, we consider how plans, and especially plan steps – which are sequences of actions – can be added to our reasoner, and show how the implementation in [Devanbu & Litman, 1991] can be reproduced in our framework.

2 DL Reasoners

In addition to defining classes of individuals, descriptions can be used for a number of purposes, such as abstracting commonalities of other sets of descriptions, providing "descriptive" answers rather than just enumerations of values, and ascribing incomplete information to individuals. For example, we can state that some procedure being planned, `NewSort`, has at least two parameters, by ascribing to it the term `at-least(2,params)`, and we can do so without having to know yet their exact identity. (See [Borgida 1992b] for a survey of DL applications.)

DL-based systems perform several different kinds of reasoning using descriptions: determining whether one description subsumes another, whether two descriptions are mutually disjoint, classifying a new description in the hierarchy of previously defined ones, testing individuals for membership in concepts, etc.

These kinds of reasoning are based on the subsumption relationship between descriptions (written as `C` $\Longrightarrow$ `D`), and the recognition of individuals by descriptions (written `b` $\longrightarrow$ `C`). It will also be convenient to introduce a derived judgment between descriptions, called equivalence ($\vdash C \equiv D$), which is defined as $\vdash C \Longrightarrow D$ and $\vdash D \Longrightarrow C$. In this paper we concentrate exclusively on subsumption reasoning, though our other work also considers reasoning with individuals.

The key advantage of DLs over First Order Predicate Calculus is the syntactic differentiation of special kinds of assertions and the computational efficiency which comes from special purpose procedures being used instead of general theorem proving. The following syntax is the starting point of the family of DLs implemented by the PROTODL system [Borgida & Brachman, 1992]:

```
<Description> ::=
        <Obj-Desc>
      | NOTHING              ;; the inconsistent description
<Obj-Desc> ::=
        <class identifier>
      | and(<Obj-Desc>+)                    ;; intersection
      | ANY-OBJECT           ;; the class of all objects
<role> ::= <identifier> | <attribute>
<attribute> ::= <identifier>
```

One can therefore see PROTODL as a system which knows about conjunction, and how to expand definitions, i.e., it knows about inheritance.

We will introduce the techniques for specifying and implementing extensions to PROTODL through a simple example: suppose that we had already extended the above system to handle the constructors **prim**, **all** and **at-least**[2], and we now want to add the **at-most** constructor.

2.1 Specification of DL reasoners

The intended meaning of **at-most(4,params)** is that it can be consistently applied only to individuals that have no more than 4 fillers for the **params** role. This can be expressed formally in the familiar notation of denotational semantics. We suggest however an additional presentation, using inference rules in the natural-deduction style. Their general form is illustrated by $\frac{\vdash c \Longrightarrow d}{\vdash \mathbf{all}(p,c) \Longrightarrow \mathbf{all}(p,d)}$, a rule describing the inference that more restrictive ranges for roles create more specific descriptions. (Here p is a variable of type role, and c and d are variables of type description.) The denominator — part below the line — of a rule is a *sequent* of the form $\vdash \alpha \Longrightarrow \beta$, expressing the judgment that description β subsumes α, while the numerator of the rule is a set of other sequents or general conditions relating to the variables appearing in them.

The rules of inference involving **at-most** are presented in Figure 1. These rules are fairly obvious; for example, MAX0-ALL0 indicates that requiring all the fillers of a role to be members of the inconsistent/empty concept is equivalent to requiring zero role fillers, while isa-MAX says that a smaller limit on the number of fillers is more restrictive.

2.2 Extending the PROTODL implementation

The PROTODL software architecture was designed to facilitate adding new description constructors, without undue loss of efficiency in implementation. PROTODL, along with many other DL implementations, has chosen to support a particular approach to reasoning with DLs: pre-computing inferences at the time information is told to it (e.g., "normalizing and pre-classifying descriptions") so that when questions are asked, the answering process is relatively straightforward and fast.[3] Therefore, description processing has

any-MAX: $\vdash$ ANY-OBJECT $\equiv$ **at-most**(∞,p)

noth-MAX: $\dfrac{n > m}{\vdash \mathbf{and}(\mathbf{at\text{-}most}(m,p), \mathbf{at\text{-}least}(n,p)) \equiv \text{NOTHING}}$

isa-MAX: $\dfrac{n < m}{\vdash \mathbf{at\text{-}most}(n,p) \Longrightarrow \mathbf{at\text{-}most}(m,p)}$

conj-MAX: $\vdash$ **and**(**at-most**(n,p),**at-most**(k,p)) $\equiv$ **at-most**(*min*(n,k),p)

MAX0-ALL0: **at-most**(0,p) $\equiv$ **all**(p,NOTHING)

Figure 1: Inference rules for **at-most**

several steps: **Parsing** — (parse tree) $\longrightarrow$ **Normalization** — (normalized storage structure) $\longrightarrow$ **Classification** — (lowest existing subsumers) $\longrightarrow$ **ExtentDetermination** — (individuals satisfying this description) $\longrightarrow$ **RecordKeeping**. To extend PROTODL with a new constructor, K say, the "programmer" is required to decide on data structures for storing information about parsed and normalized descriptions built using K, and provide five new procedures — *Parse_K*, *Normalize_K*, *Conjoin_K*, *Subsumes_K*, *Recognizes_K* — which will be incorporated into the first four stages above.

In the case of **at-most**, we will want to add an extra integer field to the record structure holding the **at-least** and **all** restrictions of a role; we also add a couple of functions, *Add_ATMOST* and *Get_ATMOST*, to access this field. Parsing is straightforward because our prefix operator syntax leads to a simple recursive descent parser.

Subsumption reasoning — the focus of this paper — is implemented using the functions *Normalize_K*, *Conjoin_K* and *Subsume_K*, based on the notion of normalization: converting descriptions to a form in which "structural subsumption" can apply. The idea of structural subsumption is that in considering two descriptions that are conjunctions of many sub-terms, it is only necessary to compare for subsumption sub-terms built with the same constructor[4]. For this reason, in the case of the constructor **at-most**, we would for example expand out any occurrence of **at-most**(0,p) or **all**(p,NOTHING) to **and**(**at-most**(0,p), **all**(p,NOTHING)) so that we do not have to look at **all**-restrictions when checking for

[2] A primitive description, **prim**(C), corresponds to a natural kind in the sense that it has no necessary and sufficient conditions; the other constructors were illustrated in our earlier example.

[3] As it turns out, PROTODL can also be used if some inferences are left to be done at question-answering time, but this issue will not be discussed here.

[4] Note again that PROTODL does not enforce this approach – it is only a heuristic that we have found to work well in almost all the cases we have tried.

```
function Conjoin_K(T,This) =

;; conjoins the new term K(T) onto the already-normalized description This

old := [get_K](This) ;; find the part of This dealing with constructor K
if old ≠ nil
then if (T occurs syntactically in old) or [SubsumesSame_K](T,old)
     then signal redundant exception
     else ;; obtain a possibly stronger constructor by combining T with old
                T := [ConjoinToSame_K](T,old)

if [SubsumesDifferent_K](T,This) ;; T is implied by other constructors in This
then signal redundant exception
else T:= [ConjoinToDifferent_K](T,This) ;; obtain a possibly stronger T
     if type(T) ≠ K ;; the stronger description might involve another constructor than K
     then post T to ImplicationsQueue ;; for later processing
     else if [Inconsistent_K](T,This) ;; T is inconsistent with the rest of This
          then signal inconsistent exception
          else [Add_K](T,This) ;; add the description to the normalized descriptor This
               [FindOtherImplications_K](T,This,ImplicationsQueue)
               ;; add any additional constructors implied by T and This.
```

Figure 2: Standard implementation of Conjoin_K

at-most-restrictions. Conversely, normalization eliminates obviously redundant subterms, as in the case when we have two **at-most**-constraints on some role.

The normalization of a description and the expansion of abbreviations introduced by defined names is driven by two pre-defined procedures for the built-in constructor **and**, namely *Normalize_AND* and *Conjoin_AND*. The basic algorithm of *Normalize_AND(C)*, for example, consists of repeatedly normalizing each subterm, built with constructor K say, and then adding its information to the new concept being built up:

```
ND := copy of ANY-OBJECT
    ;; a concept with no restrictions yet
For every subterm K(T) of C do
  T-norm := Normalize_K(T);
  Conjoin_K(T-norm,ND);
return(ND)
```

In the case of terms built with the constructor **at-most** (e.g., **at-most(2,params)**), *Normalize_ATMOST* is trivial, since the argument — the number 2 — is not composite, and parsing has already detected errors like negative integers and undefined roles.

Suppose we want to conjoin the term **at-most(2,params)** to a normalized description ND. If ND already has an **at-most** restriction on that role, the new one may be implied by the old one (e.g., **at-most(1,params)**), in which case the new one can simply be discarded as redundant; or the new one can be combined with the old one, taking the minimum of the values. We may have to repeat this process with other constructors: an **all(params,NOTHING)** restriction present in ND means that we really should have **at-most(0,params)**. Moreover, we must check if the addition of the new construct would make the concept inconsistent: in this case, verify that there is no **at-least(n,params)** with n higher than 2. Finally, we must consider the possibility that adding this constructor may imply some other one (e.g., adding **at-most(0,params)** requires adding **all(params,NOTHING)**).

By analyzing the normalization algorithms for a large number of constructors we have arrived at the more refined understanding of the *Conjoin_K* procedure shown in Figure 2, which uses half-a-dozen other, smaller procedures. The purpose of these is to make the process of constructing the implementation more systematic and less error prone. In some ways, such procedures resemble the "protocol of inference" introduced in [Rowley *et al*, 1986].

Note that in the case of **at-most**, although not necessarily for all constructors, the various subprocedures needed can be easily seen as implementing various rules of inference:

- *SubsumesSame_ATMOST* checks that the integer bound of the subsumer is higher, according to rule **isa-MAX**.
- *ConjoinToSame_ATMOST* implements rule conj-

MAX, by returning the *min* of the two numbers.

- *SubsumesDifferent_ATMOST* and *ConjoinToDifferent_ATMOST* are not needed because an **all**(p,**nothing**) restriction will take care of posting an **at-most**(0,p) constraint (see below). For this reason, the corresponding code can be edited out from *Conjoin_ATMOST*.
- *Inconsistent_ATMOST* checks if there is a higher **at-least** bound, as per rule noth-MAX.
- *FindOtherImplications_ATMOST* checks, according to rule MAX0-ALL0, to see if a there is a NOTHING **all**-constraint present when the **at-most**-bound is 0; if not, it adds such a term to be **and**-ed to the concept.

As we remarked above, when adding the **at-most** constructor to PROTODL, we will have to go back to the implementation of the **all** constructor, and modify one of its procedures, namely *FindOtherImplications_ALL*, so that it posts an **at-most**(0,p) constraint to the list of new additions whenever an **all**(p,NOTHING) is encountered.

The procedure *Conjoin_K* and its components, are used in several places in the PROTODL implementation. For example, the procedure *Subsume_K* has the following standard implementation, re-using subprocedures *SubsumesSame_K* and *SubsumesDifferent_K*:

```
function Subsumes_K(T,lowerConcept) =

  if SubsumesSame_K(T,get_K(lowerConcept))
  then return true
  else
    if SubsumesDifferent_K(T,lowerConcept)
    then return true
    else return false
```

For K=**at-most** we therefore have that *Subsume_ATMOST* is just a macro call to *SubsumeSame_ATMOST*.

3 Revisiting CLASP

Suppose that PROTODL has already been extended (using the techniques described above) to a language DL_0, with such standard constructors as **prim**, **all**, **at-least**, etc.

In turn, CLASP is intended to extend DL_0 by managing information about states, actions, and plans. We will first examine how CLASP's state and action descriptions are added to DL_0, and then turn to the more complex case of plan expressions.

Throughout, we follow two simple heuristics:

> (i) Consider first what *individuals* look like, before considering *descriptions* and their denotation. (ii) Introduce new description constructors (even domain specific ones), when the ground language does not perform all desired inferences.

3.1 CLASP states and actions

CLASP adopts a propositional representation for states of the world: a particular situation s is described by a collection of propositional symbols $\{p, q\}$, such as $\{$*phone1-off-hook, phone2-ringing*$\}$). This approach is modeled by creating individuals s for world states, and providing for every propositional symbol p a primitive concept P. Then proposition p holds in s iff s is a known instance of concept P. For example, phone1 is ringing in the world represented by the individual state s iff s is an instance of the (primitive) concept `Phone1-is-Ringing`, which is a subconcept of the special (domain-specific) concept ANY-STATE. (It will be useful to have an operation *Parents* such that *Parents*(s) returns the set of concepts of which the individual s is an instance.) Although it is possible to consider more elaborate state descriptions (e.g., ones involving parameters), we adopt here the original approach in order to concentrate on the other aspects of the extension.

An individual action then connects two states: the start and end ones [5]. CLASP chose to represent this as a DL_0 object, with two attributes: `start` and `end`. An action concept must then be subsumed by **and**(**all**(start,ANY-STATE), **all**(end,ANY-STATE)). Generic actions (like `MakePhone1Ring`) have pre- and post-conditions, which are just **all**-restrictions on the states filling the `start` and `end` attribute respectively. In order to do traditional planning, actions need also to record "delete" and "add" lists — which are (conjoined) states in this case. CLASP does this through two more attributes, `add` and `delete`. The individual fillers of these are irrelevant for individual actions. Therefore, an action concept such as `MakePhone1Ring`, might be described by the DL_0 concept

and(**prim**(ACTION),
  all(start, Phone1-is-Functioning),
  all(end, Phone1-is-Ringing),
  all(add, Phone1-is-Ringing),
  all(delete, Phone1-is-Not-Ringing))

So far, everything is modeled in DL_0. However, there is further semantics to actions: the end state's propositions are obtained from the start state's through the familiar STRIPS add/delete process: for every individual action y we have *Parents*(y.`end`) = *Parents*(y.`start`) + *Parents*(y.`add`) − *Parents*(y.`delete`).

[5] Since actors do not figure in our inferences, we ignore them henceforth.

any-ACT	$\vdash$ ANY-ACT $\equiv$ act(ANY-STATE,ANY-STATE, ANY-STATE,ANY-STATE))	Definition of ANY-ACT
noth1-ACT	$\dfrac{\vdash i \Longrightarrow \text{NOTHING}}{\vdash \text{act}(i,\_,\_,\_) \Longrightarrow \text{NOTHING}}$	Pre-conditions cannot be false
noth2-ACT	$\dfrac{\vdash e \Longrightarrow \text{NOTHING}}{\vdash \text{act}(\_,e,\_,\_) \Longrightarrow \text{NOTHING}}$	Post conditions cannot be false
noth3-ACT	$\dfrac{\vdash a \Longrightarrow \text{NOTHING}}{\vdash \text{act}(\_,\_,a,\_) \Longrightarrow \text{NOTHING}}$	Add-lists cannot be false
noth4-ACT	$\dfrac{\vdash e \Longrightarrow x\ ,\ \vdash d \Longrightarrow x\ ,\ \not\vdash \text{ANY-STATE} \Longrightarrow x}{\vdash \text{act}(\_,e,\_,d) \Longrightarrow \text{NOTHING}}$	Delete list and post-condition cannot overlap
isa-ACT	$\dfrac{\vdash i \Longrightarrow i'\ ,\ \vdash e \Longrightarrow e'\ ,\ \vdash a \Longrightarrow a'\ ,\ \vdash d \Longrightarrow d'}{\vdash \text{act}(i,e,a,d) \Longrightarrow \text{act}(i',e',a',d')}$	This is the definition used in CLASP.
conj-ACT	$\vdash$ and(act(s,e,a,d), act(s',e',a',d')) $\equiv$ act(and(s,s'),and(e,e'),and(a,a'),and(d,d'))	
norm-ACT	$\vdash$ act(_,e,a,_) $\equiv$ act(_,and(e,a),a,_)	Add-list strengthens the postcondition

Figure 3: Axioms defining the semantics of act

While CLASP did reason in this way about individuals, inferences can also be drawn about action concepts. For example, certain combinations of all-restrictions are incoherent: if the delete list includes a part of the post condition (e.g., and(... all(end,C),all(delete,C))), or the postcondition and the add-list are mutually inconsistent (e.g., and(all(end, at-most(2, rings)), all(add, at-least(3,rings))). It is however impossible to express in DL_0 conditions which would make such combinations of all-restrictions incoherent. We choose therefore to extend the language with new constructor(s), for which additional, specialized inferences can be performed. One possibility is to introduce a 4-place constructor, act, whose four arguments represent the pre, post, add and delete state descriptions. The definition of MakePhone1Ring in this new notation would be act(Phone1-is-Functioning, Phone1-is-Ringing, Phone1-is-Ringing, Phone1-is-Not-Ringing).

3.2 Specifying the semantics of actions

Based on experience with both the specification and implementation of DLs, we suggest the following heuristics in writing down rules expressing the semantics of a new constructor K (heuristics which have in fact been followed in Figure 1 for K=at-most):

> (1) Identify (or introduce) some concept that corresponds to the top of its subsumption hierarchy, and give any equivalences with it. (2) Identify the circumstances under which the constructor is involved in an inconsistent description – one denoting the empty set. (3) Define the so-called structural subsumption rule(s) for K: when does K(α) subsume K(β)? (4) Define the result of conjoining K(α) to K(β). (5) Give inferences drawn from the combination of K with each of the other term constructors; look particularly for subsumption and conjunction rules; preferably, present rules as manipulations of a description to a normal form where structural subsumption can apply.

Applying this strategy to act, we get the axioms in Figure 3. Note that if we wanted to maintain a connection between the new kinds of actions and ordinary DL_0 attributes (so that action individuals will still have start and end states associated via attributes), we can add an axiom, attribs-ACT, that mandates a connection between these features:

$\vdash$ and(act(s,e,a,d),all(start,s'),all(end,e')) $\equiv$ act(and(s,s'),and(e,e'),a,d)

3.3 Implementing actions in PROTODL

As discussed earlier, we must provide implementations of procedures *Normalize_ACT*, *Conjoin_ACT* and *Subsumes_ACT*, with the last two in fact being composed from smaller procedures. As in the case of the **at-most** constructor, the functions written for **act** correlate well with the rules of inference given.

Normalize_ACT recursively invokes *Normalize_AND* on its four arguments, and then replaces the tuple (s, e, a, d) by $(s, Conjoin\_AND(e, a), a, d)$, as per rule norm-ACT. The result is a 4-tuple to be stored in the extended data structure for normalised descriptions.

Procedure *Inconsistent_ACT* checks if the least common subsumer[6] of e and d is more specific than ANY-STATE (see rule noth4-ACT), while *SubsumesSame_ACT* and *ConjoinToSame_ACT* implement rules *isa-ACT* and *conj-ACT* respectively.

ConjoinToDifferent_ACT needs to look for **all** restrictions on the special attributes **start** and **end**, which then need to be incorporated as per rule attribs-ACT. Conversely, *FindOtherImplications_ACT* uses the arguments of **act** to strengthen the corresponding **all** restrictions.

Conjoin_ACT follows the implementation in Figure 2, except that since *SubsumesDifferent_ACT* is empty in this case, the corresponding IF-THEN is eliminated. Such minor editing of *Conjoin_K* improves efficiency.

Finally, rule attribs-ACT indicates that when new **all**-restrictions are found on attributes such as **start**, their effect on **act** will have to be taken into account. For this reason, one of the existing procedure for **all**, namely *FindOtherImplications_ALL* has to be modified accordingly.

Note the relatively simple and systematic approach to building the implementation of **act** based on the rules of inference we suggested. In fact, one can argue that every rule of inference can be implemented in some procedure provided for by PROTODL. Moreover, all but one of the description constructors in CLASSIC have efficient implementations that directly follow the rules of inference in the same way as for **act**. Current research considers the issue of automatically generating these procedures from restricted rules of inference.

4 Plans and action chains

A particular execution of a plan (a plan individual instance) is also an ordinary DL_0 object, carrying three pieces of information: the initial and goal states of the "execution" of the plan, and the chain of states leading from one to another. Three attributes, **initial**, **goal** and **steps**, record this information. Steps however are not regular object individuals — they are sequences of action objects called *scenarios* in [Devanbu & Litman, 1991], and these are a key contribution of CLASP.

Since scenarios are no longer regular objects *à la* DL_0 – they are *sequences* – we must extend the language at the top level to deal with them: <Description>::=<Scenario-Desc>, rather than <Obj-Desc>::=<Scenario-Desc>.

Classes of scenarios (called "plan expressions" in [Devanbu & Litman, 1991]) are described using a regular expression-like notation, which we represent as follows:

<Scenario-Desc> ::= **single**(<Act-Desc>)
| **seq**(<Scenario-Desc>,<Scenario-Desc>)
| **altern**(<Scenario-Desc>,<Scenario-Desc>)
| **loop**(<Scenario-Desc>)

The constructor **single** takes as argument a single action description and denotes the set of action individuals that are its instances[7]; **seq** and **altern** represent sequencing ("concatenation") and alternation ("union") respectively, while **loop** will indicate repetition one or more times[8]. Therefore **seq(single(Dial),loop(Ring))** represents the class of scenarios consisting of one dialing action followed by one or more ringing actions, where for every action in a scenario, its final state is the initial state of the succeeding action.

Once again, we introduce a top for the hierarchy, ANY-ACT-SEQ, and present in Figure 4 rules of inference for the new constructors, grouped into the categories suggested by our heuristics.

The rules for expressing equivalences of action sequence concepts are more numerous, and include rules for stating that constructor **seq** is associative, that **altern** is both associative and commutative, as well as the following:

$$\vdash \mathbf{seq}(\alpha,\mathbf{altern}(\beta,\gamma)) \equiv \mathbf{altern}(\mathbf{seq}(\alpha,\beta),\mathbf{seq}(\alpha,\gamma))$$

$$\vdash \mathbf{seq}(\mathbf{altern}(\beta,\gamma),\alpha) \equiv \mathbf{altern}(\mathbf{seq}(\beta,\alpha),\mathbf{seq}(\gamma,\alpha))$$

$$\vdash \mathbf{loop}(\alpha) \equiv \mathbf{altern}(\mathbf{seq}(\mathbf{loop}(\alpha),\alpha),\alpha)$$

$$\vdash \mathbf{loop}(\alpha) \equiv \mathbf{altern}(\mathbf{seq}(\alpha,\mathbf{loop}(\alpha)),\alpha)$$

$$\frac{\vdash \alpha \equiv \mathbf{altern}(\mathbf{seq}(\alpha,\beta),\gamma)}{\vdash \alpha \equiv \mathbf{altern}(\mathbf{seq}(\gamma,\mathbf{loop}(\beta)),\gamma)}$$

[6] This function is potentially useful for many other purposes in a DL reasoner [Cohen *et al*, 1992].

[7] The original CLASP language did not have a special constructor for this case, but we use abstract syntax here, rather than surface syntax.

[8] Axiomatisation for zero or more times is just more tedious.

	single	seq	altern	loop
any_K				$\vdash$ any-act-seq $\equiv$ loop(single(any-act))
noth_K	$\frac{\alpha \Longrightarrow \text{nothing}}{\text{single}(\alpha) \Longrightarrow \text{nothing}}$	$\frac{\vdash \alpha \Longrightarrow \text{nothing}}{\vdash \text{seq}(\alpha,\beta) \Longrightarrow \text{nothing}}$ $\frac{\vdash \beta \Longrightarrow \text{nothing}}{\vdash \text{seq}(\alpha,\beta) \Longrightarrow \text{nothing}}$	$\frac{\alpha \Longrightarrow \text{nothing}}{\text{altern}(\alpha,\beta) \Longrightarrow \beta}$	$\frac{\alpha \Longrightarrow \text{nothing}}{\text{loop}(\alpha) \Longrightarrow \text{nothing}}$
isa_K	$\frac{\alpha \Longrightarrow \beta}{\text{single}(\alpha) \Longrightarrow \text{single}(\beta)}$	$\frac{\vdash \alpha \Longrightarrow \alpha' \ , \ \vdash \beta \Longrightarrow \beta'}{\vdash \text{seq}(\alpha,\beta) \Longrightarrow \text{seq}(\alpha',\beta')}$	$\frac{\vdash \alpha \Longrightarrow \gamma \ , \ \vdash \beta \Longrightarrow \gamma}{\vdash \text{altern}(\alpha,\beta) \Longrightarrow \gamma}$ $\frac{\vdash \alpha \Longrightarrow \beta}{\vdash \alpha \Longrightarrow \text{altern}(\beta,\gamma)}$	$\frac{\alpha \Longrightarrow \alpha'}{\text{loop}(\alpha) \Longrightarrow \text{loop}(\alpha')}$
conj_K	$\vdash$ and(single(α),single(α')) $\equiv$ single(and(α,α'))	$\vdash$ and(seq(single(α),β)), seq(single(α'),β') $\equiv$ seq(single(and(α,α'), and(β,β')))	$\vdash$ and(altern(α,β),γ) $\equiv$ altern(and(α,γ), and(β,γ))	

Figure 4: Four classes of rules for plan expression constructors

This set of axioms is sufficiently complicated that we would be happy to have something like the following result

Theorem 1 *The rules presented in Figure 4 are a sound and complete axiomatization for subsumption of arbitrary object-sequence (not just action-sequence) concepts, when the operators* **seq**, **and**, **altern** *and* **loop** *are interpreted, in the natural way, as concatenation, set intersection, set union, and Klecne-closure without null sequence.*

In fact, this theorem can be proven using results from the field of Formal Languages [Salomaa 1991]. We do remark however that although such completeness proofs are very desirable and provide insights, they are usually very difficult because non-standard proof techniques are required in the absence of negation.

The above semantics is however still incomplete for actions: it ignores the fact that for two individual actions to be executed in sequence we need the end state of the first one to be the same as the start state of the second action, in the sense that the same propositions hold in each state. For this reason we need to add the following rule of inference, ACT-SEQ:

$$\vdash \text{seq}(\text{single}(\text{act}(\text{-},\text{e1},\text{-},\text{-})),\text{single}(\text{act}(\text{s2},\text{-},\text{-},\text{-}))) \equiv \text{seq}(\text{single}(\text{act}(\text{-},\text{and}(\text{e1},\text{s2}),\text{-},\text{-})), \text{single}(\text{act}(\text{and}(\text{s2},\text{e1}),\text{-},\text{-},\text{-})))$$

This rule can then be used together with the various equivalence rules to propagate information to actions throughout the plan. Among others, this can lead to discovering incoherent plans, as in the case of a sequence of actions where the start state of the second action is inconsistent with the end state of the first action. (We remark that this aspect of the semantics of action sequences was not considered in [Devanbu & Litman, 1991].)

4.1 Implementing plan expressions

In contrast to **act**, the implementation of plan expression descriptions is much more complicated, due to the fact that there is no known normal form for regular expressions. Instead, the original implementors of CLASP used their knowledge of data structures and computer science to construct a kind of deterministic finite automaton that represents regular expressions of actions, and then reasoned with it. We now proceed to show how this implementation of plan expressions can be realized in PROTODL.

One could try to construct the deterministic automaton directly, but it is handled more efficiently by building recursively from the regular expression a *nondeterministic* automaton (one with null-transitions), and then making it deterministic in one single final pass.

The implementations of *Normalize_SINGLE, Normalize_SEQ, Normalize_ALTERN* and *Normalize_LOOP* therefore build up a non-deterministic automaton with null-transitions, according to a standard algorithm, such as the one presented in [Aho *et al*, 1986]. For example, *Normalize_SINGLE*(**single**(*A*)) returns two states connected by a single transition, labeled by *Normalize_ACT(A)*.

Since all other reasoning will be done with deterministic finite automata, the remaining procedures (such as *Conjoin_SINGLE, Conjoin_SEQ*, ... , *Subsume_SINGLE*, etc.) are not needed in this case.

To actually perform the needed inferences, we augment the abstract syntax of the language with another constructor, **top-plan-exp**, which dominates the topmost plan expression constructor, and we put all the work in the implementation of **top-plan-exp**:

Normalize_TOPEXP uses the above *Normalize* functions to create the non-deterministic finite automaton, and then exactly the algorithm described in [Devanbu & Litman, 1991] is used to make it "deterministic". In addition, according to rule ACT-SEQ, we need to perform the following for each node in the automaton: conjoin to the post-condition of the action on every incoming arc the disjunction of the pre-conditions occurring on the outgoing arcs; and conversely, for the pre-conditions of outgoing arcs.

Inconsistent_TOPEXP eliminates arcs with inconsistent actions on them. If as a result no final state is reachable, the plan expression is inconsistent.

SubsumesSame_TOPEXP behaves as in [Devanbu & Litman, 1991]: it computes the intersection automaton and checks for the absence of (`Non-final,Final`) states.

ConjoinToSame_TOPEXP, which was also absent in [Devanbu & Litman, 1991], just builds the intersection deterministic finite automaton.

SubsumesDifferent_TOPEXP, *ConjoinToDifferent_TOPEXP* and *FindOtherImplications_TOPEXP* are all left empty.

This completes the implementation of plan expression reasoning for subsumption. We can now turn to the specification and implementations of plans themselves — remember that plans have plan expressions as steps; since these pose no additional novel problems, we omit this excercise.

Note again that the implementation of plan expression reasoning presented here is not novel: except for some additional inconsistency checking, it is taken from [Devanbu & Litman, 1991]; nor was it made any easier by the use of PROTODL. In fact, the significance of this exercise lies exactly in demonstrating that the PROTODL framework, while apparently quite restrictive, is sufficiently powerful to accommodate a very unusual implementation technique, which was proposed entirely independently of the PROTODL development effort.

5 Conclusions and Caveats

Let us revisit the claims made at the beginning of this paper:

(1) *Focusing on the extension of DLs through the addition of new constructors.*
The heuristic of adding new constructors led us to **act**, which performed new inferences regarding the interaction of post-conditions and add/delete lists. It also provides additional advantages: in the original CLASP, the subsumption relation between actions is determined by the subsumption rules of DL_0: an action is more specialised iff its pre and post-conditions are more specialized, as are its add and delete lists. This is however by no means the only possible arrangement. For example, in object-oriented specifications and languages, such as Eiffel [Meyer 1988], the more specialized action needs to have a precondition that is no stricter, so that it can be invoked anywhere the more general action could be invoked. This approach could easily be taken in our reconstruction by using the followings subsumption rule for **act**:

$$\frac{\vdash i \Longrightarrow i' \ , \ \vdash e \Longrightarrow e' \ , \ \vdash a \Longrightarrow a' \ , \ \vdash d \Longrightarrow d'}{\vdash \mathbf{act}(i',e,a,d) \Longrightarrow \mathbf{act}(i,e',a',d')}$$

and implementing it appropriately in *SubsumesSame_ACT*. In contrast, there would seem to be no way to encode action concepts in DL_0/CLASSIC while still obtaining this "contravariant" subsumption rule.

Similarly, it can be argued that a more specialized action could have a *less restrictive* delete-list: since it is agreed that more specialized actions have stronger post-conditions, then, all other things being equal, having fewer propositions deleted makes the post-condition stronger. Once again, the new constructor could be implemented with this new semantics, while the semantics of subsumption for **all** in DL_0 could not be altered just for the role `delete`

A second advantage of extending the language, rather than building a new system *on top of* an existing DL-reasoner as done in [Devanbu & Litman, 1991, Heinsohn 1992, Weida & Litman, 1992], is that the resulting system is "orthogonal": the new kinds of descriptions can be combined with the old ones, as role restrictions for example.

(2) *Natural semantics specifications.*
For any language, it is beneficial to have alternative formalisations, which can be proven equivalent (see [Hoare & Lauer, 1974]). Axiomatic specifications appear to have the advantage (not demonstrated here, but shown in [Borgida 1992a]) of describing incomplete reasoners more easily than denotational semantics. The rules of inference presented in this paper are "natural", elegant and concise. (For comparison, we invite the reader to give corresponding semantics for CLASP in First Order Logic, say.) Compared to other axiomatic approaches, they have advantages such as the existence of PL&C tools like Typol [Despeyroux 1984] for rapid prototyping in Prolog, and the suitability of natural deduction proofs for explanation (see [Borgida 1991]).

(3) *The* PROTODL *system.*
While elsewhere we show how constructors in CLASSIC can be added to PROTODL, here we showed how

PROTODL can be used to implement the subsumption-reasoning of CLASP. Note that the original CLASP system was built *as a layer on top of* CLASSIC, rather than by modifying the CLASSIC implementation itself. Presumably, CLASSIC's implementation was too daunting to modify. CLASP could have been added to most other existing DLs, but we strongly suspect that they would not have been any easier to modify. We emphasize that our extension performs the same efficient, custom-made actions (ignoring the "improvements" we made — see below) as originally proposed in [Devanbu & Litman, 1991]: there is no meta-interpreter to slow things down, and we are in fact not paying any measurable additional run-time price for having used PROTODL. Note also that this was achieved by the simple addition of two *abstract syntax constructors*, for **single** and **top-plan-exp**. As detailed in [Borgida & Brachman, 1992], the PROTODL system comes with built-in algorithms of general use in DL processing, including implementations of inheritance and classification algorithms, efficient management of large transitive is-a and disjointness hierarchies, symbol table management, error reporting, etc. Presumably, the implementors of CLASP had to reproduce most of these for their extension.

The present works differs from most earlier work on extensible or modifiable reasoners [Greiner & Lenat, 1980, Genesereth 1983, Lenat & Guha, 1990, Rowley *et al*, 1986] by focusing on DLs (and thereby gaining considerable leverage); and it differs from KRS [Gaines 1991] (which admittedly considers many other issues), by PROTODL's much more detailed system architecture, its emphasis on constructing efficient reasoners, the connection with formal specifications, and especially the systematization described below.

(4) *Systematization*: The heuristic classification of our rules was suggested by the different kinds of procedures that PROTODL asks the implementor to provide or modify. These often correlate well with implementations, as seen for **act**, but there must be room for imaginative work, as shown for plan expressions. The advantages of our systematic approach of building a system from components are supported by the following improvements over the original CLASP proposal:

(a) The original CLASP system appears to assume that users of the language will not be defining **inconsistent** actions, plans, etc. While they may not do so intentionally, experience with building large knowledge bases indicates that it is essential to check for such conditions. During the search for rules of equivalence with NOTHING, we have identified a number of additional cases where actions and plan expressions can be inconsistent, such as those expressed in rules noth4-ACT, norm-ACT. The search also led us to consider the more general inference rule ACT-SEQ, dealing with action chains.

(b) By explicitly thinking about the interaction of **act** and **all** we were led to discover the inference which led to rule norm-ACT.

(c) Plans are regular DL_0 concepts; therefore plans can be conjoined with **and**. This involves conjoining the restrictions on the attributes `initial`,`goal` and `steps`, with the restriction on `steps` being a plan expression. Therefore, we need to be able to conjoin plan expression! Because CLASP does not have an explicit **and** operator applicable to plan expressions, the original paper did not consider this operation. On the other hand, we came up with the idea when seeking to implement *Conjoin_TOPEXP*.

In conclusion, by reconstructing, and incidentally improving upon, the innovative and significant work on CLASP, we claim to have provided evidence for the thesis that our approach helps develop new description logics and their implementations, through the process of extending PROTODL, in a systematic and less error-prone way. This work cannot and does not make any sweeping claims to generality: there are other approaches to implementing efficient DL reasoners, (e.g., [MacGregor 1990, Donini *et al*, 1991]); and we have no proof that one cannot build bad implementations using PROTODL, nor that all good implementations can be easily built with it.

There remains of course a lot of work to be done, especially concerning the extension of PROTODL using new role-constructors.

Acknowledgments

I am deeply grateful to Ron Brachman for giving me the opportunity to join him in the CLASSIC adventure, and for his collaboration on PROTODL as well as other applications, which made apparent the need for this research. Discussions with Richard Fikes and Jeffrey Ullman, and the comments of Prem Devanbu and Diane Litman have, I hope, improved the arguments and presentation of this work – they were most generous with their time. This work was funded in part by NSF Grant IRI-9119310.

References

[Aho *et al*, 1986] Aho, A., R. Sethi, and J. Ullman, *Compilers: Principles, Techniques, and Tools*, Addison-Wesley, 1986

[Borgida 1991] Borgida, A., "Terminologic Frames as Types: Inference rules and prospective applications", Technical Report, Department of Computer Science, Rutgers University, May 1991.

[Borgida 1992a] Borgida, A., "From Type Systems to Knowledge Representation: Natural Semantics

Specifications for Description Logics," *Int. J. of Intelligent and Cooperative Information Systems* 1(1), June 1992.

[Borgida 1992b] Borgida, A., "A new look at Description Logics and their Utility in Information Management (or Description Logics are not just for the Flightless-Birds)", Technical Report, Rutgers University, July 1992.

[Borgida *et al*, 1989] Borgida, A., Brachman, R. J., McGuinness, D. L., and L.A. Resnick, "CLASSIC: A Structural Data Model for Objects," *Proc. 1989 ACM SIGMOD International Conference on Management of Data*, June, 1989, pp. 59–67.

[Borgida & Brachman, 1992] Borgida, A., and R.J. Brachman, "ProtoDL: A Customizable Knowledge Base Management System", *Conf. on Information and Knowledge Management*, Baltimore, November 1992.

[Cohen *et al*, 1992] Cohen, W., A. Borgida and H. Hirsch, "Computing Least Common Subsumers in Description Logics", *Proc. AAAI'92* (San Jose), July 1992.

[Despeyroux 1984] Despeyroux, T., "Executable specification of static semantics", *Semantics of Data Types*, LNCS Vol. 173, June 1984.

[Devanbu & Litman, 1991] Devanbu, P., and D. Litman, "Plan-based Terminological Reasoning," *Proc. KR'91*, Boston, MA, 1991.

[Devanbu *et al*, 1991] Devanbu, P., Brachman, R. J., Ballard, B. W., and P.G. Selfridge, "LaSSIE: A Knowledge-Based Software Information System," *Communications of the ACM*, 34(5), May, 1991.

[Donini *et al*, 1991] Donini, F., Lenzerini, M., Nardi, D., and W. Nutt, "Tractable Concept Languages", *Proc. IJCAI'91*, Australia, August 1991, pp. 458-463.

[Doyle & Patil, 1991] Doyle, J, and R. Patil, "Two theses of knowledge representation: language restrictions, taxonomic classification, and the utility of representation services", *Artificial Intelligence 48*(3), April 1991, pp.261–298.

[Heinsohn 1992] Heinsohn, J., D. Kudenko, B. Nebel, H.J. Profitlich, *RAT: Representation of actions using terminological logics*, DFKI, University of Saarlandes, Germany, 1992.

[Gaines 1991] Gaines, B.R., "Empirical investigation of knowledge representation servers: design issues and applications experience with KRS", *ACM SIGART Bulletin* 2(3), June 1991.

[Genesereth 1983] Genesereth, M., "An overview of Meta-Level Architecture," *Proc. AAAI-83*, Washington, DC, 1983.

[Greiner & Lenat, 1980] Greiner, R., and D. Lenat, "RLL: A Representation Language Language," *Proc. AAAI-80*, Stanford, CA, 1980.

[Hoare & Lauer, 1974] Hoare, C.A.R., and P.E. Lauer, "Consistent and complementary formal theories of the semantics of programming languages", *Acta Informatica*, 3(2), 1974, 135–154

[Kahn 1988] Kahn, G. "Natural Semantics", Rapport de Recherche No. 601, INRIA, Sophia Antipolis, France.

[Lenat & Guha, 1990] Lenat, D., and R. Guha, *Building Large Knowledge-Based Systems*. Addison Wesley, 1990.

[MacGregor 1990] MacGregor, R., "The Evolving Technology of Classification-based Knowledge Representation Systems", in *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, John Sowa editor,Morgan-Kaufman 1990.

[Meyer 1988] Meyer, B. *"Object Oriented Software Construction"*, Prentice Hall, 1988.

[Rowley *et al*, 1986] Rowley, S., Shrobe, H., Cassels, R. "Joshua: Uniform access to heterogeneous knowledge structures", *AAAI'86*, pp.48-52, 1986

[Salomaa 1991] Salomaa, A., *Jewels of Formal Language Theory*, Computer Science Press, 1981.

[Schmiedel, 1990] Schmiedel, A., "A Temporal Terminologic Reasoner", in *Proceedings AAAI-90*, Boston, MA, August 1990, 641–645.

[Weida & Litman, 1992] Weida, R., and D. Litman, "Terminological Reasoning with Constraint Networks and an Application to Plan Recognition", *Third Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'92)*, Cambridge, MA., 1992.

[Woods & Schmolze, 1992] Woods, W.A., and J.G. Schmolze, "The KL-ONE family", *Computers and Mathematics with Applications 23*(2-5), 1992.

An Empirical Analysis of Optimization Techniques for Terminological Representation Systems

or: Making KRIS get a move on

Franz Baader, Bernhard Hollunder,
Bernhard Nebel, Hans-Jürgen Profitlich
German Research Center for AI (DFKI)
Stuhlsatzenhausweg 3, 6600 Saarbrücken 11, Germany
e-mail: (*last name*)@dfki.uni-sb.de

Enrico Franconi
Istituto per la Ricerca
Scientifica e Tecnologica (IRST)
38050 Povo TN, Italy
e-mail: franconi@irst.it

Abstract

We consider different methods of optimizing the classification process of terminological representation systems, and evaluate their effect on three different types of test data. Though these techniques can probably be found in many existing systems, until now there has been no coherent description of these techniques and their impact on the performance of a system. One goal of this paper is to make such a description available for future implementors of terminological systems. Building the optimizations that came off best into the KRIS system greatly enhanced its efficiency.

1 INTRODUCTION

Terminological representation systems can be used to represent the taxonomic and conceptual knowledge of a problem domain in a structured and well-formed way. To describe this kind of knowledge, one starts with atomic concepts (unary predicates) and roles (binary predicates), and defines more complex concepts and roles using the operations provided by the concept language of the particular formalism. In addition to this concept description formalism, most terminological representation systems also have an assertional component, which can be used to express facts about a concrete world.

Of course, it is not enough to have a system that just stores concept definitions and assertional facts. The system must also be able to reason about this knowledge. An important inference capability of a terminological representation system is *classification*. The classifier computes all *subsumption* relationships between concepts, i.e., the subconcept-superconcept relationships induced by the concept definitions. In this paper we consider only optimizations for the classification process. We do not take into account problems that are specific to assertional reasoning. This concentration on the terminological component is partially justified by the fact that this is the part that partakes in most reasoning activities of almost all systems—which means that the efficiency of this reasoning component is crucial for the overall behavior of the system.

The first terminological representation system, KL-ONE [Brachman and Schmolze, 1985], was an implementation of Brachman's work on structured inheritance networks [Brachman, 1977]. In the last decade many knowledge representation systems based on these ideas have been built, for example BACK [Peltason, 1991], CLASSIC [Patel-Schneider *et al.*, 1991], KANDOR [Patel-Schneider, 1984], KL-TWO [Vilain, 1985], K-Rep [Mays *et al.*, 1991], KRYPTON [Brachman *et al.*, 1985], KRIS [Baader and Hollunder, 1991], LOOM [MacGregor, 1991], MESON [Edelmann and Owsnicki, 1986], NIKL [Schmolze and Mark, 1991], SB-ONE [Kobsa, 1991], and YAK [Cattoni and Franconi, 1990]. Moreover, formal aspects of terminological representation languages have been thoroughly investigated, with the highest emphasis having been placed on the decidability and complexity of the subsumption problem (see, e.g., [Levesque and Brachman, 1987; Nebel, 1988; Schmidt-Schauß, 1989; Patel-Schneider, 1989; Nebel, 1990b; Schmidt-Schauß and Smolka, 1991; Donini *et al.*, 1991a; Donini *et al.*, 1991b]). As a result of these investigations, it is known that subsumption determination is at least NP-hard or even undecidable for reasonably expressive languages. The developers of terminological representation systems usually have reacted to this problem in one of the following two ways. On the one hand, there are systems such as CLASSIC which support only a very limited terminological language, but employ almost complete reasoning methods. On the other hand, systems such as LOOM provide for a very powerful language, but the reasoning is incomplete, which means that not all existing subsumption relationships are detected.

The only system that does not make this compromise, i.e., that provides complete algorithms for a very expressive concept description language, is KRIS. Obviously, this means that KRIS will need exponential

time for worst case examples which, on the one hand, are not expressible in the less expressive systems, and which are, on the other hand, treated more efficiently, but less completely, by systems with fast and incomplete algorithms. However, it is not *a priori* clear whether this also implies that KRIS has to be less efficient for "typical" knowledge bases. In particular, it might at least be fast in cases where its full expressive power is not used, or where incomplete algorithms are still complete. The empirical analysis of terminological representation systems described in [Heinsohn *et al.*, 1992] seems to preclude this possibility, though. KRIS turned out to be much slower than, for example, CLASSIC, even for knowledge bases that are in the scope of CLASSIC's concept language, and for which CLASSIC's subsumption algorithm is complete.

One aim of the present paper is to demonstrate that this bad performance of KRIS is not mainly due to the use of complete subsumption algorithms, but instead to the fact that the tested version was the first implementation of an experimental system where efficiency considerations only played a minor role. For this purpose we shall consider possible optimizations of the classification process on three different levels. The optimizations on the *highest level* are independent of the fact that what we are comparing are concepts defined by a terminological language. On this level, classification is considered as the abstract order-theoretic problem of computing a complete representation of a partial ordering (in our case the subsumption hierarchy) by making as few as possible explicit comparisons (in our case calls of the subsumption algorithm) between elements of the underlying set (in our case the set of all concepts occurring in the terminology). Optimizations on the *next level* still leave the subsumption algorithm unchanged, but they do employ the fact that we are not comparing abstract objects but instead structured concepts. At this level subsumption relationships that are obvious consequences of this structure can be derived without invoking the subsumption algorithm. On the *third level*, the actual subsumption algorithm is changed so that it can benefit from the information provided by subsumption relationships which have previously been computed.

The effects these optimizations have on the classification process are evaluated on three different sets of test data. As in [Heinsohn *et al.*, 1992] we consider both existing knowledge bases used in other projects and randomly generated knowledge bases whose structure resembles those of the real knowledge bases. Since the first level of optimizations can be done in an abstract order-theoretic setting, these optimizations are also evaluated on randomly generated partial orderings (see [Winkler, 1985] for a description of the generation process we have used).

It should be noted that we do not claim that all the presented optimizations are novel. Most optimizations can probably be found in many of the existing systems (see e.g. [Lipkis, 1982; MacGregor, 1988; Peltason *et al.*, 1989; Woods, 1991; Ellis, 1991]). However, until now it was not possible to find a coherent description of all of them, and there were no empirical studies on their exact effects. A second motivation for this work is to make such a description available for future implementors of terminological representation systems.

2 COMPUTING THE SUBSUMPTION HIERARCHY

In the first level of optimizations we are concerned with computing the concept hierarchy induced by the subsumption relation. More abstractly, this task can be viewed as computing the representation of a partial ordering. For a given partial ordering[1] $\leq$ on some set P, $\prec$ shall denote the *precedence relation* of $\leq$, i.e., $\prec$ is the smallest relation such that its reflexive, transitive closure is identical with $\leq$. Obviously, $x \prec y$ iff $x \leq y$ and there is no z different from x and y such that $x \leq z \leq y$. If $x \leq y$, we say that x is a *successor* of y and y is a *predecessor* of x. Similarly, if $x \prec y$, we say that x is an *immediate successor* of y and y is an *immediate predecessor* of x.

Given a set X and a partial ordering $\leq$ on X, computing the representation of this ordering on X amounts to identifying $\prec$ on X. (If $\leq$ is a *total* ordering, this task is usually called sorting.) The basic assumption here is that the partial ordering is given via a comparison procedure, and that the comparison operation is rather expensive. For this reason, the complexity of different methods to compute the precedence relation is measured by counting the number of comparisons. Of course, the number of other operations should not be too high as well.

In our case, X is the set of concepts defined in a terminological knowledge base, and $\leq$ is the subsumption relation between these concepts. The assumption that the subsumption test is the most expensive operation is justified by the known complexity results for the subsumption problem [Donini *et al.*, 1991a]. To be more precise, the subsumption relation is only a quasi-ordering, i.e., it need not be antisymmetric. For the following discussion, this is mostly irrelevant, however. There is only one place in the algorithms where this fact has to be taken into account.

The worst case complexity of computing the representation of a partial ordering on a set with n elements is obviously $O(n^2)$ because it takes $n \times (n-1)$ comparisons to verify that a set of n incomparable elements is indeed a flat partial order. Since subsumption hierarchies typically do not have such a "pathological"

[1] A partial ordering is a transitive, reflexive, and antisymmetric relation.

structure, considerably less than $n \times (n-1)$ comparisons will almost always suffice.

Below, we describe and analyze four different methods to identify the representation of a partial ordering, namely, the *brute force* method, the *simple traversal* method, the *enhanced traversal* method, and the *chain inserting* method. Average case analyses of these methods seem to be out of reach since one does not know enough about the structure of "typical" terminological knowledge bases, and since it is not even known how many different partial orders exist for a given number of elements [Aigner, 1988]. For this reason, the different methods are compared empirically.

All methods we describe are incremental, i.e., assuming that we have identified the precedence relation $\prec_i$ for $X_i \subseteq X$, the methods compute for some element $c \in X - X_i$ the precedence relation $\prec_{i+1}$ on $X_{i+1} = X_i \cup \{c\}$. The two most important parts of this task are the *top search* and the *bottom search*. The top search identifies the *set of immediate predecessors* in X_i for a given element c, i.e., the set $X_i{\downarrow}c := \{x \in X_i \mid c \prec x\}$. Symmetrically, the bottom search identifies the *set of immediate successors* of c, denoted by $X_i{\uparrow}c$.

To be more precise, the procedures for top search that we will describe below compute the set $\{x \in X_i \mid c \leq x \text{ and } c \not\leq y \text{ for all } y \prec_i x\}$, which in most cases is the set $X_i{\downarrow}c$. Because the subsumption relation is only a quasi-ordering, there is one exception. The concept c can be equivalent to an element x of X_i, i.e., $c \leq x$ and $x \leq c$. In this case, the top search procedures will yield $\{x\}$ instead of $X_i{\downarrow}c$. To take care of this case, we test $x \leq c$ whenever the top search procedure yields a singleton set $\{x\}$. If this test is positive, c is equivalent to x, and we know that $X_i{\downarrow}c = X_i{\downarrow}x$, and $X_i{\uparrow}c = X_i{\uparrow}x$, which means that we don't need the bottom search phase. Otherwise, the result of the top search procedure is in fact $X_i{\downarrow}c$.

Given the set $X_i{\downarrow}c$, $X_i{\uparrow}c$, and $\prec_i$, it is possible to compute the precedence relation $\prec_{i+1}$ on $X_{i+1} = X_i \cup \{c\}$ in linear time. In fact, one just has to add $\prec$-links between c and each element of $X_i{\downarrow}c$, and between each element of $X_i{\uparrow}c$ and c. In addition, all $\prec$-links between elements of $X_i{\uparrow}c$ and $X_i{\downarrow}c$ have to be erased.

2.1 THE BRUTE FORCE METHOD

The top search part of the brute force method can be described as follows:

1. Test $c \leq x$ for all $x \in X_i$.
2. $X_i{\downarrow}c$ is the set of all $x \in X_i$ such that the test succeeded and for all $y \prec_i x$ the test failed.

The bottom search is done in the dual way.

This method obviously uses $2 \times |X_i|$ comparisons for the step of inserting c in X_i. Summing over all steps leads to $n \times (n-1)$ comparison operations to compute the representation of a partial ordering for n elements. Further, this is not only the worst-case, but also the best-case complexity of this method.

2.2 THE SIMPLE TRAVERSAL METHOD

It is obvious that many of the comparison operations in the brute force method can be avoided. Instead of testing the new element c blindly with all elements in X_i, in the top search phase the partial ordering can be traversed top-down and in the bottom search phase bottom-up, stopping when immediate predecessors or successors have been found. This leads us to the specification of the simple traversal method.

The top search starts at the top[2] of the already computed hierarchy. For each concept $x \in X_i$ under consideration it determines whether x has an immediate successor y satisfying $c \leq y$. If there are such successors, they are considered as well. Otherwise, x is added to the result list of the top search.

In order to avoid multiple visits of elements of X_i and multiple comparisons of the same element with c, the top search algorithm described in Figure 1 employs one label to indicate whether a concept has been "visited" or not and another label to indicate whether the subsumption test was "positive," "negative," or has not yet been made. The procedure *top-search* gets two concepts as input: the concept c, which has to be inserted, and an element x of X_i, which is currently under consideration. For this concept x we already know that $c \leq x$, and *top-search* looks at its direct successors with respect to $\prec_i$. Initially, the procedure is called with $x = \top$. For each direct successor y of x we have to check whether it subsumes c. This is done in the procedure *simple-top-subs?*. Since our hierarchy need not be a tree, y may already have been checked before, in which case we have memorized the result of the test, and thus need not invoke the expensive subsumption procedure *subs?*. The direct successors for which the test was positive are collected in a list *Pos-Succ*. If this list remains empty, x is added to the result list; otherwise *top-search* is called for each positive successor, but only if this concept has not been visited before along another path.

The bottom search can be done again in the dual way. It is interesting to note that this top search is in principle the same as the one described by Lipkis [Lipkis, 1982], who implemented the first classification algorithm for KL-ONE. The bottom search described by Lipkis, however, is more efficient than the one given here.

[2] We assume that our concept hierarchies always contain a top element $\top$ and a bottom element $\bot$.

```
top-search(c,x) =
    mark(x,"visited")
    for all y with y ≺_i x do
        if simple-top-subs?(y,c)
          then Pos-Succ ← Pos-Succ ∪ {y}
        fi
    od
    if Pos-Succ is empty
      then Result ← {x}
      else for all y ∈ Pos-Succ do
             if not marked?(y,"visited")
               then Result ← Result ∪ top-search(c,y)
             fi
           od
    fi
```

```
simple-top-subs?(y,c) =
    if marked?(y,"positive")
      then Result ← true
      elsif marked?(y,"negative")
           then Result ← false
           elsif subs?(y,c)
                then mark(y,"positive")
                     Result ← true
                else mark(y,"negative")
                     Result ← false
           fi
    fi
```

Figure 1: Top search phase of the simple traversal method

2.3 THE ENHANCED TRAVERSAL METHOD

Although the simple traversal method is a big advantage compared with the brute force method (see Figure 5 (a)), it still does not exploit all possible information. First, during the top search phase, we can take advantage of tests that have already been performed. Second, in the bottom search phase, we can use the information gained during the top search as well.

Of course, a dual strategy is also possible, i.e., performing the bottom search before the top search and exploiting the information gathered during the bottom search phase. Analyzing Figure 5, it becomes quickly obvious that this strategy would be less efficient, however. In fact, for the simple traversal method—where the top and bottom phase are done in a symmetric way—the top search phase turns out to be a lot faster. Thus it is better to start with this phase because the information gained thereby can then be used to speed up the slower bottom search phase.

When trying to take advantage of tests that have already been performed during top search one can either concentrate on negative information (i.e., that a subsumption test did not succeed) or on positive information (i.e., that a subsumption test was successful).

To *use negative information* during the top search phase one has to check whether for some predecessor z of y the test $c \leq z$ has failed. In this case, we can conclude that $c \not\leq y$ without performing the expensive subsumption test [MacGregor, 1988]. In order to gain maximum advantage, all predecessors of y should have been tested before the test is performed on y. This can be achieved by using a modified breadth-first search where the already computed hierarchy is traversed in topological order, as described by Ellis [1991]. Alternatively, one can make a recursive call whenever there is a predecessor that has not yet been tested. This is what the procedure *enhanced-top-subs?* described in Figure 2 does. If we replace the call of *simple-top-subs?* in *top-search* by a call of *enhanced-top-subs?*, we get the top search part of the enhanced traversal method.

```
enhanced-top-subs?(y,c) =
    if marked?(y,"positive")
      then Result ← true
      elsif marked?(y,"negative")
           then Result ← false
           elsif for all z with y ≺_i z
                       enhanced-top-subs?(z,c)
                 and   subs?(y,c)
                 then mark(y,"positive")
                      Result ← true
                 else mark(y,"negative")
                      Result ← false
           fi
    fi
```

Figure 2: Top search phase of the enhanced traversal method. The procedure *top-search* is adopted from the simple traversal method, but instead of *simple-top-subs?* it calls *enhanced-top-subs?*

The enhanced top search procedure just described makes maximum use of failed tests. Alternatively, it is possible to *use positive information*. Before checking $c \leq y$, one can look for successors z of y that have passed the test $c \leq z$ [MacGregor, 1988]. If there exists such a successor, one can conclude that $c \leq y$ without performing an actual subsumption test. Although we are only interested in minimizing the number of comparison operations, it should be noted that

instead of searching for a successor that has passed the test it is more efficient to propagate positive information up through the subsumption hierarchy. This can be achieved by an easy modification of the procedure *simple-top-subs?*. When the call "subs?(y,c)" yields *true*, not only y is marked "positive," but so are all of y's predecessors. Obviously, this technique cannot be combined with the enhanced top search described in Figure 2 since it reduces the number of subsumption tests only if there are predecessors which have not yet been tested, and enhanced top search tests all predecessors before making a subsumption test.

Neither of these alternatives is uniformly better than the other one, which can be seen by considering the examples described in Figure 3 and 4.

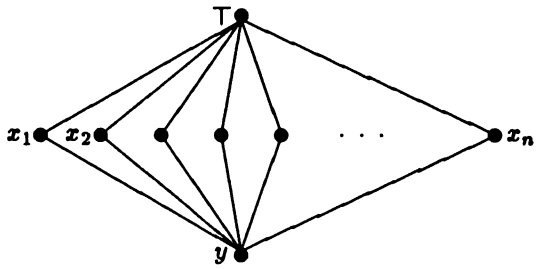


Figure 3: The new element c is a direct successor of y

In the first example, the top-search using negative information makes $n+1$ tests: it first tests x_1, then goes to y, but before testing it, it tests all its direct predecessors, i.e., $x_2, \ldots, x_n$. The top search using positive information makes two tests: first x_1 and then y; the positive result of this second test is propagated to $x_2, \ldots, x_n$.

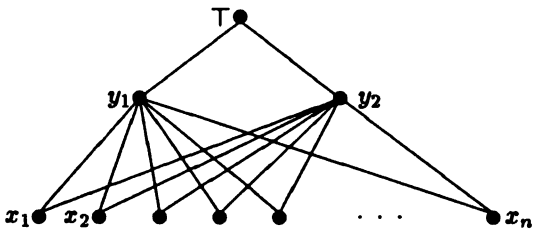


Figure 4: The new element c is a direct successor of y_1, but not a successor of y_2, $x_1, \ldots, x_n$

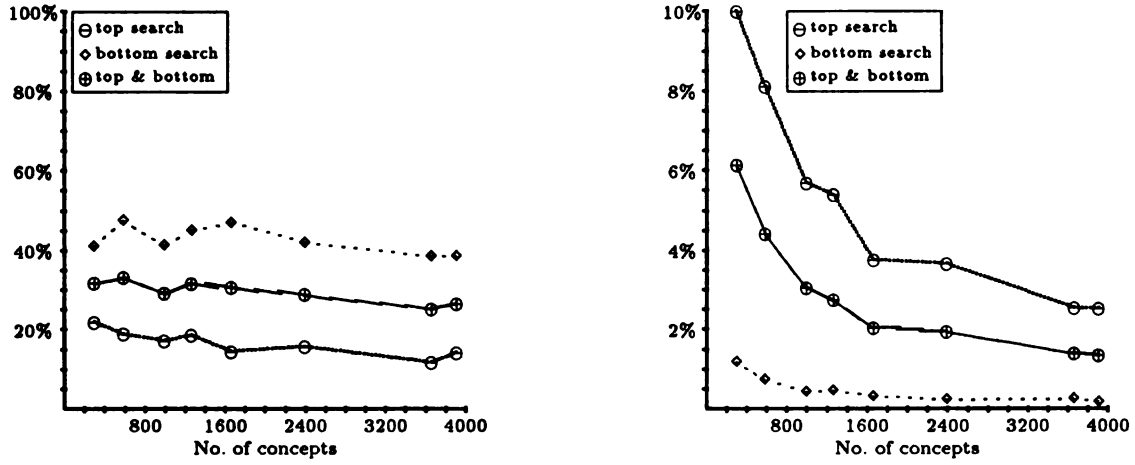
In the second example, the top search using negative information needs only two tests: first it tests y_1, then goes to x_1, but before testing x_1 its direct predecessor y_2 is tested. The negative result of this test prevents $x_1, \ldots, x_n$ from being tested. The top search using positive information tests $n+2$ nodes: first y_1, then all its successors $x_1, \ldots, x_n$, and finally y_2.

However, we have observed significant performance differences for the two different top search strategies. For the random knowledge bases, the method using positive information was only slightly better than the simple traversal method (less than 5%). For this reason, we have also considered a "hybrid method" which propagates positive information up, and negative information down the hierarchy (but does not test all predecessors before testing a node). Propagating negative information down is again achieved by an easy modification of *simple-top-subs?*. When the call of "subs?(y,c)" yields *false*, not only y is marked "negative," but all of y's successors. The hybrid method turned out to be a lot better than just propagating positive information, but it still needed slightly more tests (approx. 5%–10%) than the enhanced top search for all but one of the random knowledge bases. On five of the six realistic knowledge bases the "hybrid method" was insignificantly faster than the enhanced top search (less than 1%). On the remaining realistic KB, the hybrid method needed 10% more comparisons. Although these results do not seem to be conclusive in favor of the "hybrid method" or the enhanced top search, it is obvious that the use of negative information leads to a significantly greater reduction of comparisons than the use of positive information.

Now we turn to the *bottom search* phase of the enhanced traversal method. Of course, optimizations dual to the ones described for the top search can be employed here. In addition, the set $X_i{\downarrow}c$ can be used to severely cut down the number of comparisons in the bottom search phase. As mentioned by Lipkis [1982], the search for immediate successors of c can be restricted to the set of successors of $X_i{\downarrow}c$. In fact, the set of candidates for $X_i{\uparrow}c$ is even more constrained. Only elements that are successors of *all* $x \in X_i{\downarrow}c$ can be immediate successors of c [Ellis, 1991]. This optimization is achieved by an easy modification of the procedure *enhanced-bottom-search* (which is dual to *enhanced-top-search*): the test "marked?(y,"negative")" is augmented to "marked?(y,"negative") or y is not a successor of all $x \in X_i{\downarrow}c$." The remaining problem is how to implement the second part of this test. One possibility is to mark the successors of the elements of $X_i{\downarrow}c$ in an appropriate way, and then test these labels. Another possibility, which we have used in our tests, is to equip each concept in X_i with a list of all its predecessors in X_i, and test whether $X_i{\downarrow}c$ is contained in the list of predecessors of y.

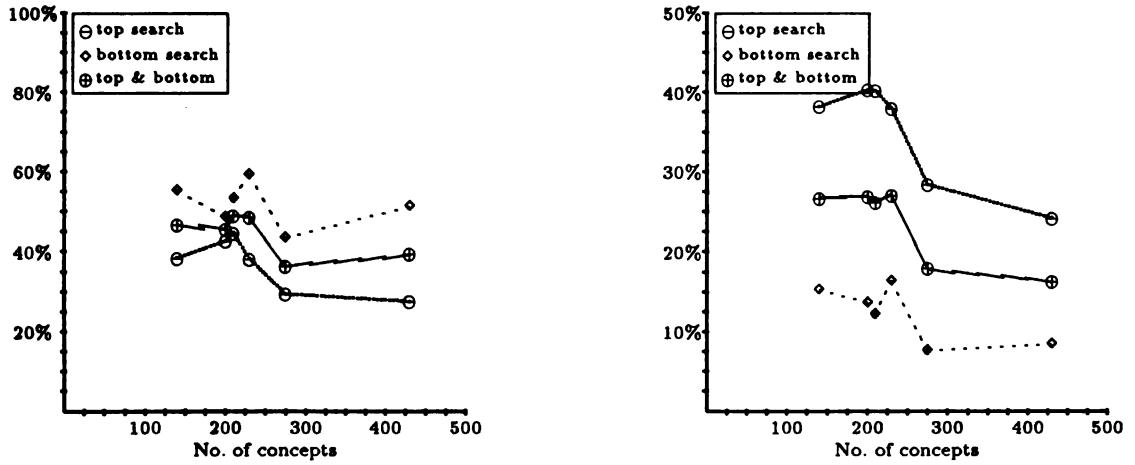
As a result of this optimization, the number of necessary comparison operations can be cut down to a fraction compared with the simple bottom search strategy. Interestingly, we observed a further reduction of comparison operations in case of the real knowledge bases when searching top-down starting at $X_i{\downarrow}c$ instead of searching bottom-up. For the random knowledge bases, no such difference was observed, however.

The effects of the simple and enhanced traversal



(a) Simple traversal method (b) Enhanced traversal method

Figure 5: Number of comparison operations relative to *brute force* method for random knowledge bases



(a) Simple traversal method (b) Enhanced traversal method

Figure 6: Number of comparison operations relative to *brute force* method for realistic knowledge bases

method for the random knowledge bases and the realisitc knowledge base as test data are displayed in Figures 5 and 6. These graphs present the number of necessary comparisons *relative* to the brute force method for the top search and the bottom search phase, as well as for the entire classification process.

2.4 THE CHAIN INSERTING METHOD

Sorting a set of elements that is linearly ordered can be either done by incrementally searching the already ordered sequence linearly or by using binary search. In the former case, we inevitably end up with quadratic complexity, while in the latter case $O(n \times \log n)$ is a possibility. Of course, it seems attractive to transfer the latter technique to our problem, an idea that leads to the chain inserting method.

In order to specify the method, we first define the notion of a chain covering of a partial ordering. A *chain covering* is a partition of a partial ordering into *chains*, i.e., totally ordered subsets. Provided we have a chain covering of the set X_i, it is possible to identify the sets $X_i{\downarrow}c$ and $X_i{\uparrow}c$ by binary search in all chains. For a given chain C_j of the covering $X_i = C_1 \cup \cdots \cup C_m$, binary search is used to find the least predecessor and the greatest successor of c in C_j. Since the underlying ordering $\leq$ is only a partial ordering on X, the new element c to be inserted into the chain C_j need not be comparable with all elements of C_j. For this reason one needs two binary search phases for each chain. The first one asks $c \leq x$, and treats negative answers as if they would mean $c > x$. This phase yields the least predecessor of c in C_j. The other phase is dual, and yields the greatest successor of c in C_j. The set

of these least predecessor (resp. greatest successors) for all chains of the covering yields a superset of $X_i\downarrow c$ (resp. $X_i\uparrow c$). The set $X_i\downarrow c$ (resp. $X_i\uparrow c$) is obtained by eliminating the elements which are not minimal (resp. maximal) with respect to $\leq_i$. As a further optimization, propagation of positive and negative information of successful and of failed tests in the existing subsumption hierarchy is used to make some of the explicit subsumption tests during binary search superfluous, after one or more chains have already been searched through.

We have also considered a "hybrid" method that employs chain inserting for long chains and enhanced traversal afterwards. The idea here is that by binary search in long chains one gets rather quickly into the "center" of the partial ordering, from which propagation of positive and negative information should have the greatest effect.

It is, of course, advisable to use chain coverings with a minimal number of chains. Unfortunately, the computation of minimal chain coverings is nontrivial and takes more than quadratic time [Jungnickel, 1990]. Nevertheless, *simple heuristics* permit the incremental construction of chain coverings that are almost optimal. The heuristic we have used to update the chain covering when a new element c is inserted proceeds as follows. After the sets $X_i\downarrow c$ and $X_i\uparrow c$ have been computed, c is inserted in the longest chain satisfying one of the following conditions:

1. Binary search has yielded both a least predecessor and greatest successor in the chain, and they are successive elements of the chain. In this case, c is inserted between these two elements in the chain.
2. Binary search has yielded a least predecessor (or greatest successor) in the chain, and it is the least (resp. greatest) element of the chain. In this case, c is inserted below (resp. above) this element in the chain.

If there is no chain satisfying one of these conditions, a new chain consisting of c is created. In our experiments, the chain coverings obtained this way were less than 10% suboptimal.

The empirical results concerning the performance of the chain inserting method are not conclusive. To our surprise, the chain inserting method turned out to be less efficient than the enhanced traversal method on the random and real knowledge bases, even though it is still a lot better than the simple traversal method. The "hybrid" version using chain inserting for long chains and enhanced traversal afterwards also turned out to be less efficient than the pure enhanced traversal method. On the other hand, for tests on randomly generated partial orders the chain inserting method in some cases showed a much better performance than the enhanced traversal method. A reason for this behavior could be that, compared to the realistic knowledge bases we used in our tests, the randomly generated partial orders have a much higher connectivity (which means that propagation of positive and negative information has more effect) and permit longer chains (which makes binary search more important). The chain inserting method may thus become more interesting for knowledge bases defining a relatively deep hierarchy. Additionally, it seems possible to exploit the chain covering in order to implement storage compression techniques as described by Jagadish [1989].

3 OBVIOUS SUBSUMPTION RELATIONSHIPS

In this section we describe some further techniques for avoiding subsumption tests by exploiting relations which are obvious when looking at the syntactic structure of concept definitions.[3] These pre-tests require only little effort but can speed up the classification process significantly. We consider three different optimizations which apply to different stages of the classification process.

The first technique can be used prior to the top search. It applies when the description of the concept c that we want to insert is conjunctive (which is the case for the majority of concepts, in particular if we consider the existing real knowledge bases). If this description mentions x explicitly as a conjunct, then it is obviously the case that $c \leq x$. We call such concepts x *told subsumers* of c. Of course, if x is also a conjunctively defined concept, it may have told subsumers as well, and these (and their told subsumers, etc.) can be included into the list of told subsumers of c. It is rather easy to compile this list while reading in the concept definitions. The information that c is subsumed by its told subsumers can be propagated through the existing hierarchy $(X_i, \prec_i)$ prior to the top search, e.g., by pre-setting the markers used in the traversal method to "positive" for the told subsumers and all their predecessors. A prerequisite for this optimization technique to be effective is that the told subsumers of c are already contained in X_i. This can be achieved by inserting concepts following the so-called "definition-order." This order can be formally defined as follows: We say that a concept x directly uses a concept y iff y occurs in the definition of x. Let "uses" be the transitive closure of "directly uses." Then x comes in the definition-order after y if x uses y.

Assuming that concepts are inserted in the subsumption hierarchy following the definition-order, another optimization can be applied. The bottom search phase can be completely avoided if a *primitive concept* (i.e., a concept that is described by giving only necessary con-

[3] These techniques are probably used in all systems, see, e.g. [Peltason *et al.*, 1989].

conditions) has to be classified. In fact, such a concept c can only subsume the bottom concept and concepts whose definitions use c. Since the second type of possible subsumees is not yet present in the actual hierarchy when inserting along the definition-order, the result of the bottom search is just the bottom concept $\bot$. Considering the fact that in realistic KBs the majority of concepts (60%-90%) are primitive, this optimization can save most of the subsumption calls during the bottom search phase. Combining the two optimization techniques led to a saving of 10% to 20% with respect to the pure enhanced traversal method for the realistic knowledge bases. In case of the random knowledge bases, the savings where even greater, as can be seen from Figure 7.

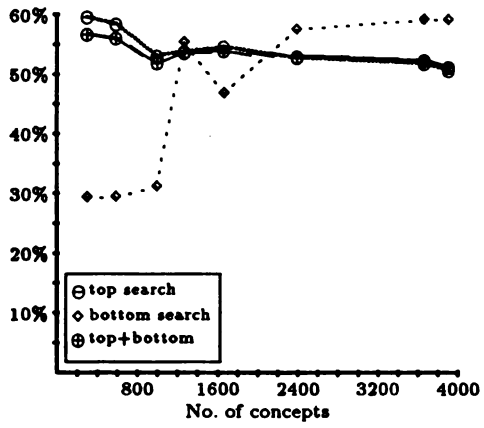


Figure 7: Number of necessary comparisons when exploiting obvious subsumption relations relative to pure *enhanced traversal* method for random KBs

A final optimization technique can be used as a pre-test before calling the subsumption algorithm. It makes use of the fact that a concept c is subsumed by a *primitive concept* x if, and only if, the completely expanded form of c contains the "primitive component of x" (see [Nebel, 1990a, p. 54–56]) as a conjunct. By extracting and caching the "primitive components" of all concepts, it becomes possible to check whether a subsumption relation is possible by comparing the sets of primitive components. If this test gives a negative result, the subsumption algorithm need not be called. Although such a test overlaps with computations the subsumption algorithm does, it is much faster than the subsumption test. For this reason, this pre-test pays off if most of the subsumption calls can be avoided, which was indeed the case for our test data. Our experiments indicate that the number of calls of the subsumption algorithm can be again reduced by 50%-60%, if this technique is applied.

4 THE SUBSUMPTION ALGORITHM

In this section we consider two possible optimizations of the subsumption algorithm, and describe the effects they have on the performance of classification for our test knowledge bases. Let us first reconsider the two types of subsumption algorithms usually implemented in terminological systems.

In almost all terminological representation systems other than KRIS *structural subsumption algorithms* are employed (e.g. CLASSIC, LOOM, BACK). Such algorithms basically proceed as follows. First, the concepts are normalized, i.e., they are transformed into equivalent normal forms. Subsumption between normalized concepts is a kind of structural comparison where each subexpression of the first concept must have a counterpart in the other concept. This algorithmic technique allows one to develop efficient subsumption algorithms which are easily shown to be sound. However, for expressive terminological languages these algorithms are usually not complete, and it is not clear how the technique could be extended in order to build complete structural subsumption algorithms.

Using a different paradigm, in the past years sound and *complete* subsumption algorithms for a large class of terminological languages have been developed (e.g. [Schmidt-Schauß and Smolka, 1991; Hollunder *et al.*, 1990]). Most of these algorithms are designed as *satisfiability checking algorithms*. These algorithms are model generation procedures, and are similar to first-order tableaux calculus, with the main difference that the specific structure of concept descriptions allows one to impose an appropriate control that ensures termination. Since a concept A subsumes a concept B if, and only if, $\neg A \sqcap B$ is not satisfiable, i.e., there does not exist an interpretation which interprets $\neg A \sqcap B$ as a non-empty set, a satisfiability algorithm in fact can be used to solve the subsumption problem. In order to check whether a given concept C is satisfiable, the tableaux-based algorithm tries to generate a finite interpretation in which C is interpreted as a non-empty set. This generation process is complete in the sense that if it fails, i.e., an obvious contradiction occurs, we can conclude that C is not satisfiable; otherwise C is satisfiable. An obvious contradiction in the model generating process occurs, for example, if some element is constrained to be both instance of a "primitive component" and its complement—which is impossible.

It is well-known that subsumption of concepts defined in a cycle-free terminology can be reduced easily to subsumption of concept terms which do not refer to other concept definitions of the terminology (so-called *expanded concept terms*) [Nebel, 1990a]. For conceptual simplicity both types of subsumption algorithms are usually described in the literature as taking expanded concept terms as arguments, which precludes

the exploitation of previously computed subsumption relationships.

4.1 THE OPTIMIZATIONS

However, almost all terminological representation systems take advantage of previously computed subsumption relationships. To illustrate how this can be done for a structural subsumption algorithm, suppose that C and D are normalized concept descriptions. As mentioned above, structural subsumption between C and D means to find for each subexpression C' of C a corresponding subexpression D' of D. Often, in turn, these subexpressions have to be tested for subsumption. In case C' and D' are concept names of possibly defined concepts, and we already know a subsumption relationship between C' and D', it is not necessary to call the subsumption algorithm recursively for (the expanded form of) C' and D'. Thus, it is rather natural and straightforward to incorporate the use of already computed subsumption relations into a structural subsumption algorithm. It should be noted that it is an essential requirement *not* to completely expand the concept definitions before checking subsumption. Further, it is necessary to classify the concepts according to the "definition-order" mentioned in the previous section.

In contrast to other terminological systems, KRIS employs a satisfiability algorithm to determine subsumption relationships between concepts. Since a satisfiability algorithm does not recursively call subsumption algorithms but satisfiability algorithms, it is not obvious how to exploit previously computed subsumption relationships. A closer look, however, reveals that a satisfiability algorithm may detect a contradiction earlier during model generation if previously computed subsumption relationships are taken into account. To see this, suppose that we already know that a defined concept A subsumes a defined concept B. If during the model generation an element is constrained to be both instance of $\neg A$ and B, a contradiction can be detected without expanding the definitions of A and B. Again, this approach only works if the concept definitions are not expanded before starting to check satisfiability.

If expansion is done "by need" during the satisfiability test, one has to decide in which order to expand the concept names. It is easy to see that this order may have considerable impact on the runtime behavior. For example, assume that we are testing $A \sqcap B$ for satisfiability where in the TBox A is defined by a very large concept description and B is defined to be $\neg A \sqcap C$. If B is expanded first, the contradiction between A and $\neg A$ is detected at once. On the other hand, if A is expanded first, detecting the contradiction between the large descriptions associated with A and its negation may be rather time-consuming, depending on the structure of the description.

One way of avoiding this problem is to expand concept names according to the inverse of their definition-order, which in the above example would mean that we expand B before A, because the definition of B refers to A. Of course, this means that for each expansion operation one has to go through the list of all expandable names, and look for a maximal one with respect to the definition-order. For our tests we have used another solution, which avoids searching for a maximal name, but may use more space. Here one expands in arbitrary order, but when a name is expanded it is not removed, but just marked as expanded. If, in our example, A is expanded before B, we then still have the name A, and as soon as B is expanded it yields the contradiction with $\neg A$.

In order to gain experience in how to optimize the satisfiability algorithm to be employed in KRIS, we implemented the following three versions.

1. The first one takes *completely expanded* concept descriptions as input. Since these descriptions do not contain names of defined concepts, obvious contradictions can only be detected between "primitive components," i.e., concept names which are not defined in the TBox.
2. The second one *successively expands* the concept descriptions during model generation, but keeps the names, as described above. This allows the algorithm to detect obvious contradictions not only between primitive components but also between names of defined concepts.
3. The third version is a refinement of the second one in that already computed subsumption relationships are taken into account when looking for obvious contradictions.

4.2 EMPIRICAL RESULTS AND ANALYSIS

It turns out that the first version is significantly slower than the second one, a result we did expect. The main reason for this behavior is that the number of recursive calls of the satisfiability algorithm is reduced due to obvious contradictions detected between names of defined concepts. As a consequence, the runtime of the second version is reduced by 40-60% relative to the first version (see Figure 8, which displays the results for the random knowledge bases).

A result we did *not* expect is that the behavior of the third version is no better than of the second, which means that trying to exploit already computed subsumption relationships does not pay off. The reason for this behavior seems to be that—at least for the test data—only a few contradictions are detected by using already computed subsumption relationships. This is indicated by the fact that the number of recursive calls of the satisfiability algorithm does not significantly de-

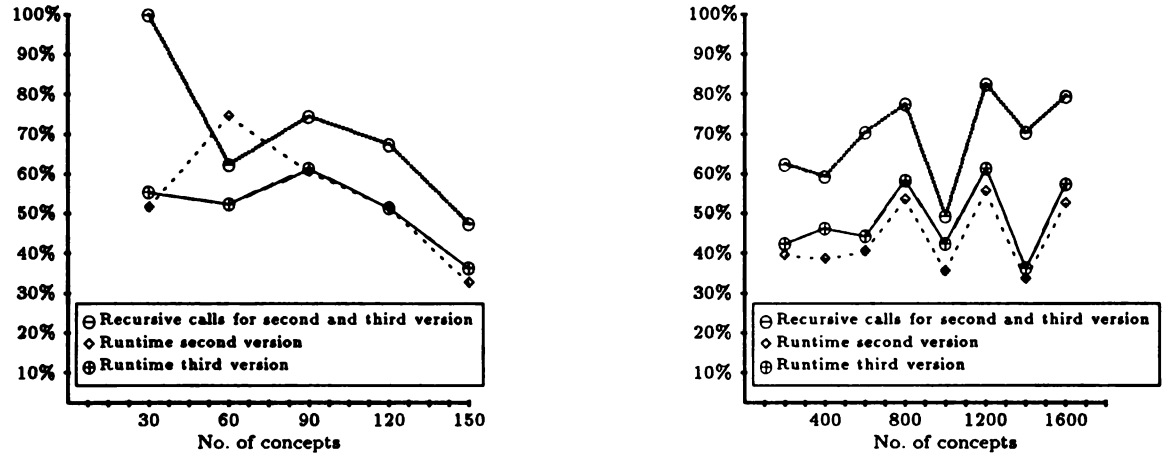


Figure 8: Runtime and number of recursive calls of the second and third version's satisfiability algorithm relative to the algorithm taking completely expanded concept terms as input (first version)

crease when going from the second to the third version. However, the test of whether a set of negated and unnegated concept names is contradictory w.r.t. already computed subsumption relationships is more complex than just searching for complementary names, which explains that the third version's runtime behavior is even slightly worse than the second one's (cf. Figure 8).

This result is all the more surprising since using computed subsumption relationships during classification is an optimization technique employed by most terminological systems. The reason why it may pay off for other systems could be that these systems first normalize, and during this normalization phase auxiliary concepts may be introduced. For example, assume that C is defined by the description $\forall R.A \sqcap \forall R.B$, and D by $\forall R.A$. The normalization procedure may introduce a new concept name E, define it as $A \sqcap B$, and modify the definition of C to $\forall R.E$. Now the subsumption relationship between A and the auxiliary concept E—which is found first if the terminology is classified according to the definition-order—immediately entails that D subsumes C. Thus classification of the terminology with the auxiliary concepts allows one to exploit previously computed subsumption relationships more often. On the other hand, it has the disadvantage that in general a lot more concepts have to be classified.

Another interesting behavior we observed is due to the interaction between different optimization techniques. The optimizations described in the previous two sections try to avoid subsumption tests, whereas the present section is concerned with speeding up the subsumption test. Ideally, one could expect that these optimizations are independent. This means that the overall speedup factor is the product of the speedup factors of the individual optimizations. This can only be true if the optimizations apply uniformly to all situations, however.

If the optimizations apply to special cases only, subsumption avoidance optimizations and subsumption test optimization may aim at similar special cases and lead to the situation that subsumption tests are avoided which have neglectable computational costs in any case.

If we take the second or third version's satisfiability algorithm, the exploitation of obvious subsumption relationships caused by *conjunctive definitions*, i.e., the first optimization technique mentioned in Section 3, does no longer speed up the classification process significantly. This is due to the fact that such subsumption relationships can now be easily detected by the satisfiability algorithms. For example, let C be a concept that is defined to be the conjunction of $C_1, \ldots, C_m$, where the C_i are defined concepts as well. The obvious subsumption relationship between C_i and C is immediately detected by the second and third version of the satisfiability algorithm, due to an obvious contradiction between C_i and $\neg C_i$.

5 CONCLUSION

We have described and analyzed different optimization techniques for the classification process in terminological representation systems. Interestingly, two of the most promising techniques, namely, the chain inserting method for computing the representation of a partial order and the exploitation of already computed subsumption relations in the subsumption algorithm, did not lead to the expected performance increase in case of realistic knowledge bases.

As a result of our empirical analysis, the optimization techniques that came off best were incorporated in the KRIS system. Whereas the unoptimized version

was orders of magnitude slower than the fastest system tested in [Heinsohn *et al.*, 1992], the new version has now a runtime behavior similar to that of the other systems on the test data used there.

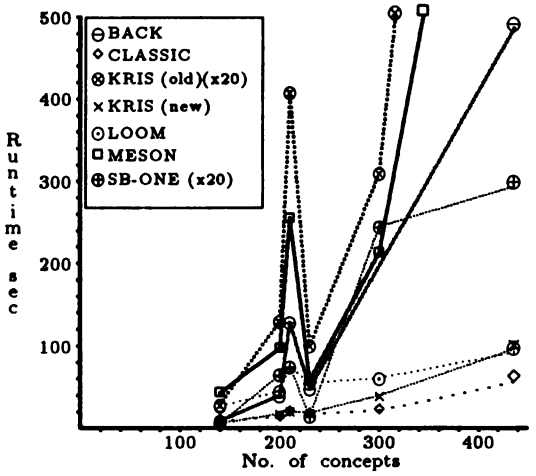


Figure 9: Runtime performance for existing knowledge bases

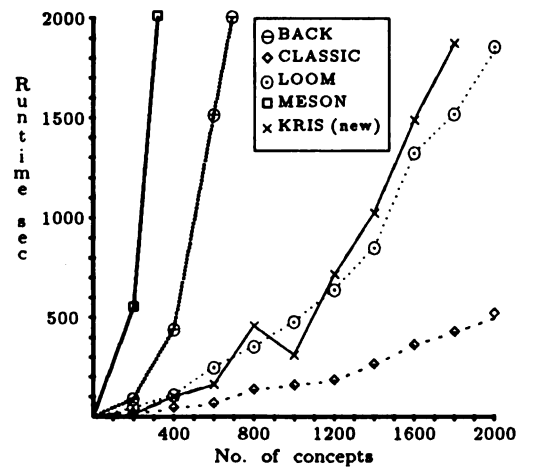


Figure 10: Runtime performance for large random knowledge bases

Figure 9 displays the runtime of the new KRIS version for the realistic knowledge bases and contrasts them with the runtime figures given in [Heinsohn *et al.*, 1992]. Figure 10 gives the results for large random knowledge bases.[4]

It should be noted, however, that all the knowledge bases used in the test are formulated using quite limited terminological languages. An interesting open problem is the development of further optimization techniques for more powerful terminological languages containing also disjunction and negation and of specific optimization techniques for assertional reasoning.

[4]The description of the runtime behavior of the systems in [Heinsohn *et al.*, 1992] refers to system versions as of 1990 and does not necessarily reflect the performance of more recent versions.

Acknowledgements

We would like to thank Uwe Utsch for implementing the different subsumption strategies, Hans-Jürgen Bürckert, Jochen Heinsohn, Armin Laux, and Werner Nutt for helpful discussions concerning the topics described in this paper, and Alex Borgida and Peter Patel-Schneider for helpful comments on an earlier version of this paper.

This work has been supported by the German Ministry for Research and Technology (BMFT) under research contracts ITW 8901 8 and ITW 8903 0 and by the Italian National Research Council (CNR), project "Sistemi Informatici e Calcolo Parallelo."

References

Martin Aigner (1988). *Combinatorical Search.* Teubner, Stuttgart, Germany.

Franz Baader and Bernhard Hollunder (1991). KRIS: Knowledge representation and inference system. *SIGART Bulletin*, 2(3):8–14.

Ronald J. Brachman and James G. Schmolze (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216.

Ronald J. Brachman, Victoria Pigman Gilbert, and Hector J. Levesque (1985). An essential hybrid reasoning system: Knowledge and symbol level accounts in KRYPTON. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 532–539, Los Angeles, CA.

Ronald J. Brachman (1977). *A Structural Paradigm for Representing Knowledge.* PhD thesis, Havard University.

Roldano Cattoni and Enrico Franconi (1990). Walking through the semantics of frame-based description languages: A case study. In *Proceedings of the Fifth International Symposium on Methodologies for Intelligent systems*, Knoxville, TN. North-Holland.

Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt (1991a). The complexity of concept languages. In James A. Allen, Richard Fikes, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference*, pages 151–162, Cambridge, MA. Morgan Kaufmann.

Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt (1991b). Tractable concept languages. In John Mylopoulos and Ray Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 458–465, Sydney, Australia. Morgan Kaufmann.

Jürgen Edelmann and Bernd Owsnicki (1986). Data models in knowledge representation systems: A case study. In Claus-Rainer Rollinger and Werner Horn, editors, *GWAI-86 und 2. Österreichische Artificial-Intelligence-Tagung*, pages 69–74, Ottenstein, Austria. Springer-Verlag.

Gerard Ellis (1991). Compiled hierarchical retrieval. In *6th Annual Conceptual Graphs Workshop*.

Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel, and Hans-Jürgen Profitlich (1992). An empirical analysis of terminological representation systems. In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence*, pages 767–773, San Jose, CA. MIT Press. An extended version of this paper is available as DFKI Research Report RR-92-16.

Bernhard Hollunder, Werner Nutt, and Manfred Schmidt-Schauß (1990). Subsumption algorithms for concept description languages. In L. C. Aiello, editor, *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 348–353, Stockholm, Sweden. Pitman.

H. V. Jagadish (1989). A compressed transitive closure technique for efficient fixed-point query processing. In L. Kerschberg, editor, *Expert Database Systems—Proceedings From the 2nd International Conference*, pages 423–446, Menlo Park, CA. Benjamin/Cummings.

Dieter Jungnickel (1990). *Graphen, Netzwerke und Algorithmen*. BI Wissenschaftsverlag, Mannheim, Germany, 2nd edition.

Alfred Kobsa (1991). First experiences with the SB-ONE knowledge representation workbench in natural-language applications. *SIGART Bulletin*, 2(3):70–76.

Hector J. Levesque and Ronald J. Brachman (1987). Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93.

Thomas Lipkis (1982). A KL-ONE classifier. In J. G. Schmolze and R. J. Brachman, editors, *Proceedings of the 1981 KL-ONE Workshop*, pages 128–145, Cambridge, MA. The proceedings have been published as BBN Report No. 4842 and Fairchild Technical Report No. 618.

Robert MacGregor (1988). A deductive pattern matcher. In *Proceedings of the 7th National Conference of the American Association for Artificial Intelligence*, pages 403–408, Saint Paul, MI.

Robert MacGregor (1991). Inside the LOOM description classifier. *SIGART Bulletin*, 2(3):88–92.

Eric Mays, Robert Dionne, and Robert Weida (1991). K-Rep system overview. *SIGART Bulletin*, 2(3):93–97.

Bernhard Nebel (1988). Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34(3):371–383.

Bernhard Nebel (1990a). *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, New York.

Bernhard Nebel (1990b). Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249.

Peter F. Patel-Schneider, Deborah L. McGuinness, Ronald J. Brachman, Lori Alperin Resnick, and Alex Borgida (1991). The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bulletin*, 2(3):108–113.

Peter F. Patel-Schneider (1984). Small can be beautiful in knowledge representation. In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, pages 11–16, Denver, Colo..

Peter F. Patel-Schneider (1989). Undecidability of subsumption in NIKL. *Artificial Intelligence*, 39(2):263–272.

Christof Peltason, Albrecht Schmiedel, Carsten Kindermann, and Joachim Quantz (1989). The BACK system revisited. KIT Report 75, Department of Computer Science, Technische Universität Berlin, Berlin, Germany.

Christof Peltason (1991). The BACK system – an overview. *SIGART Bulletin*, 2(3):114–119.

Manfred Schmidt-Schauß and Gert Smolka (1991). Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26.

Manfred Schmidt-Schauß (1989). Subsumption in KL-ONE is undecidable. In Ron J. Brachman, Hector J. Levesque, and Ray Reiter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 1st International Conference*, pages 421–431, Toronto, ON. Morgan Kaufmann.

James G. Schmolze and William S. Mark (1991). The NIKL experience. *Computational Intelligence*, 6:48–69.

Marc B. Vilain (1985). The restricted language architecture of a hybrid representation system. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 547–551, Los Angeles, CA.

P. Winkler (1985). Random orders. *Order*, 1:317–331.

William A. Woods (1991). Understanding subsumption and taxonomy: A framework for progress. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 45–94. Morgan Kaufmann, San Mateo, CA.

Terminological Reasoning with Constraint Networks and an Application to Plan Recognition

Robert Weida*and Diane Litman†
Department of Computer Science
Columbia University
New York, NY 10027

Abstract

Terminological systems, such as KL-ONE and K-Rep, are widely used in AI to represent and reason with concept descriptions. They compute subsumption relations between concepts and automatically classify concepts into a taxonomy. Each concept in the taxonomy describes a set of possible instances which are a superset of those described by its descendants. One limitation of current systems is their inability to handle complex compositions of concepts, such as constraint networks where each node is described by an associated concept. For example, plans are often represented (in part) as collections of actions related by a rich variety of temporal constraints. The T-REX system integrates terminological reasoning with constraint network reasoning to classify such plans, producing a "terminological" plan library. T-REX also introduces a new view of plan recognition as a process which dynamically partitions the plan library by modalities, e.g., necessary, possible and impossible, while actions are observed. Plan recognition is guided by the plan library's terminological nature. Although this work focuses on temporal constraint networks used to represent plans, terminological systems can be extended to encompass constraint networks in other domains as well.

1 INTRODUCTION

Terminological knowledge representation (TKR) systems support automatic classification of definitional taxonomies based on subsumption inferences. In a *definitional taxonomy*, each class describes a set of possible instances which are a superset of those described by its descendant classes. We compute subsumption (subset relationships) between classes according to the structure of their definitions, i.e., *structural subsumption*. Thus, classification via structural subsumption endows a taxonomy with formal meaning. Classification ensures that the proper location of any class within the taxonomy is uniquely determined from its definition. This in turn supports automatic detection of redundant, inconsistent and vacuous definitions. Classification also facilitates incremental construction of taxonomies, enforcement of semantics, type-checking and pattern matching. For elaboration on these benefits, see [Brachman and Schmolze, 1985, MacGregor, 1991, Woods, 1986].

While terminological systems are widely used in many application areas, to date they have focused on representing structured conceptual descriptions, or *concepts*. A critique of contemporary TKR which argues for greater expressiveness is [Doyle and Patil, 1991]. One limitation of current terminological systems, e.g., BACK [von Luck *et al.*, 1987], CLASSIC [Borgida *et al.*, 1989], K-Rep [Mays *et al.*, 1991b], and LOOM [MacGregor and Bates, 1987], is their inability to represent and reason with complex compositions of concepts, such as constraint networks where each node is described by a concept.

Plans are central to many areas of AI. In this paper we present a knowledge representation system that computes subsumption among plans represented as collections of temporally related actions. In particular, we employ a plan representation which uses temporal constraint networks in the style of [Allen, 1983]. We show how to extend the ideas of structural subsumption and classification found in TKR systems to automatically organize these plans into a definitional taxonomy which constitutes a "terminological" plan library. The advantages obtained from representing knowledge in standard terminological systems are achieved here as well. Our approach is similar in spirit to previous work

*Robert Weida is also with IBM Thomas J. Watson Research Center, Yorktown Heights, NY.

†Diane Litman is also with AT&T Bell Laboratories, Murray Hill, NJ.

on plan subsumption [Devanbu and Litman, 1991, Wellman, 1990], but provides a much richer temporal representation language. We also use our notion of constraint network subsumption to develop a new, terminological approach to plan recognition. While terminological plan systems have been applied in the areas of plan synthesis [Wellman, 1990] and plan retrieval [Devanbu and Litman, 1991], the application of terminological reasoning to the area of plan recognition has previously been unexplored.

The definitional plan taxonomy provides a natural basis to guide plan recognition. Many plan recognition systems infer agents' plans from their actions by searching libraries of possible plans for suitable (perhaps nondeductively inferred) matches. We introduce a new view of plan recognition as a process which dynamically partitions the plan library into modalities, e.g., *necessary*, *possible* and *impossible*, according to observations of the environment. We also leverage the taxonomy's enforced semantics to minimize the number of plans that must be examined. Our approach unifies representation and reasoning work in plan recognition and terminological reasoning.

A prototype plan recognition system, called T-REX, serves as a testbed for our ideas. T-REX integrates and builds upon existing systems for TKR and temporal reasoning. It represents and reasons about actions using K-Rep [Mays *et al.*, 1991a] and temporal relationships using MATS [Kautz and Ladkin, 1991]. The T-REX plan taxonomy is separate from the K-Rep concept taxonomy at present, but K-Rep's readily extensible architecture will make it easy to incorporate plans directly within the concept taxonomy.

Although we focus on temporal constraint networks used to represent plans, our methods apply to any kind of constraint network where we can define subsumption operations on the nodes and arcs, and hence on the networks themselves. We call such networks *terminological constraint networks*. In Section 6, we outline an approach to terminological reasoning with, and recognition of, N-dimensional spatial descriptions.

2 REPRESENTATION OF PLANS

Plan descriptions typically include preconditions, effects, a body composed of steps to carry out the plan, and some constraints. Following [Kautz, 1991], we will concentrate on plan recognition via plan bodies and their relationship via abstraction. We define a *plan body* as a collection of steps along with some temporal constraints between pairs of steps. Each step has a type of action associated with it. Action types are represented by generic concepts in K-Rep [Mays *et al.*, 1991a], which we shall call *action concepts*. Together, these concepts constitute an *action taxonomy*. We assume that the taxonomy includes every type of action which appears in a plan, or is observed during plan recognition. Hence, action types are considered disjoint if there is no action type that they both subsume (note that subsumption is reflexive). K-Rep also represents instantiated action concepts, or *action instances*. When there is no ambiguity, we may simply refer to action concepts and action instances as actions. Each temporal constraint is an arbitrary disjunction of Allen's exhaustive set of 13 primitive temporal relationships between intervals: *before*, *meets*, *during*, *overlaps*, *starts*, *finishes*, *equals*, and inverses of the first six relationships [Allen, 1983]. A *plan network* is a temporal constraint network [Allen, 1983] whose nodes correspond to time intervals when the steps of the plan's body occur. Hence, an action concept is associated with each node. Plans may be embedded as *macro actions* within other plans (but not within themselves). Any temporal constraint on a step with a macro action can be propagated to each substep within that macro by appropriate use of a constraint propagation algorithm such as Allen's [Allen, 1983].

Following precedent, e.g., [Kautz, 1991], we draw examples from the cooking domain. By convention, generic concept names are prefixed by C-. Names of instances are formed by concatenating a concept name with a unique number and stripping off the leading C-. All example plan descriptions in this paper will be constructed from action concepts in the taxonomy shown in Figure 1. Note that although we just use descriptive names, concepts and their instances are really represented in greater detail in K-Rep. For example, action concepts have roles such as *agent* and *object*. When a concept definition specifies necessary and sufficient conditions for class membership, K-Rep determines the concept's proper location within the taxonomy using classification. Such concepts are shown without an asterisk in Figure 1.

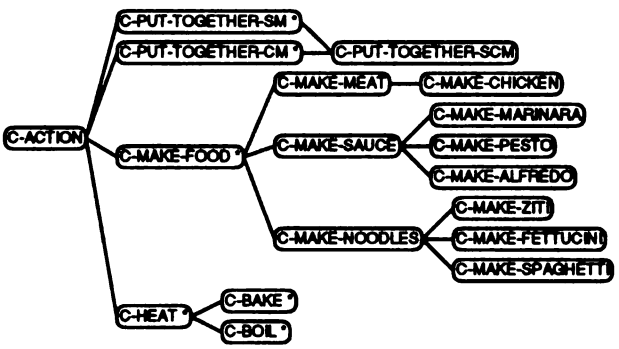


Figure 1: A Definitional Action Taxonomy

Figure 2 diagrams a simple plan to assemble chicken marinara. To simplify our diagrams, we omit trivial constraints and some other constraints which can be inferred via transitivity, and we label nodes with the names of associated action concepts. Plans denote a set of possible *plan instances*, which have bodies composed of action instances and nondisjunctive tempo-

ral constraints, chosen in accordance with the plan. When possible, we write networks as a sequence of nodes separated by constraints. The following might be an instance of the plan in Figure 2:

- MAKE-CHICKEN12 {*before*} MAKE-MARINARA54 {*before*} PUT-TOGETHER-CM27.

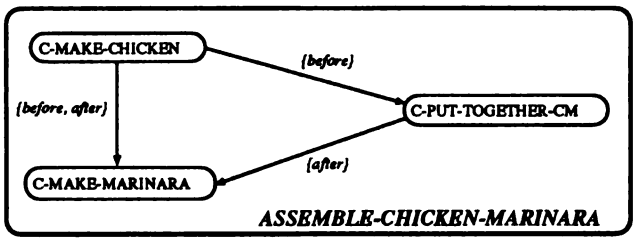


Figure 2: A Simple Plan Network

3 TERMINOLOGICAL REASONING WITH PLANS

Set theoretically, one plan *subsumes* another just in case every possible instance of the second is also an instance of the first. In this paper, we restrict our attention to inferences via plan bodies. Then structural plan subsumption can be characterized solely in terms of graph matching. Plan subsumption is based on subsumption between nodes and subsumption between arcs. We define *node subsumption* and *temporal constraint subsumption* as follows:

Definition 1 *Node* N1 *subsumes node* N2 *iff the concept associated with* N1 *subsumes the concept of* N2.

Definition 2 *Temporal constraint* C1 *subsumes temporal constraint* C2 *iff* C1*'s disjuncts are a superset of* C2*'s disjuncts.*

Clearly, whenever *C2* describes some temporal relationship then *C1* must describe it as well, e.g., *before or after* subsumes (1) *before*, and (2) *after*, as well as (3) *before or after*. For plan networks, *arc subsumption* follows immediately from temporal constraint subsumption. In other applications, we might use K-Rep concepts to represent the semantics of arcs. Structural subsumption between terminological constraint networks such as plan networks entails an appropriate mapping:

Definition 3 *A* subsumption mapping *from terminological constraint network* T1 *to terminological constraint network* T2 *maps every node* N1 *of* T1 *to a distinct node* N2 *of* T2 *such that* N1 *subsumes* N2, *and every arc between a pair of nodes in* T1 *subsumes the arc between the corresponding nodes in* T2.

In the case of plan networks, Definition 3 assumes that all nodes correspond to atomic actions, i.e., any macro actions have already been fully expanded and constraints on nodes with macro actions have been propagated to the constituents. Then we have the following theorem, proved in [Weida, 1992], which formally justifies the subsumption algorithm presented in Section 4.

Theorem 1 *Terminological constraint network T1 subsumes terminological constraint network T2 iff there exists a subsumption mapping from T1 to T2.*

The Appendix defines a plan library from which this paper's examples are drawn. Figure 3 illustrates how a HEAT-NOODLES plan network subsumes an ASSEMBLE-SPAGHETTI-MARINARA plan network (after expansion of the latter's BOIL-SPAGHETTI macro action). Dashed arrows indicate the subsumption mapping from nodes in the subsumer to nodes in the subsumee. Notice that the two plans differ in the number and specificity of their actions, as well as the specificity of the relevant constraint. This is analogous to structural subsumption in TKR, where a concept may specialize its parent(s) by further constraining their roles (constraints) and/or adding additional roles (constraints).

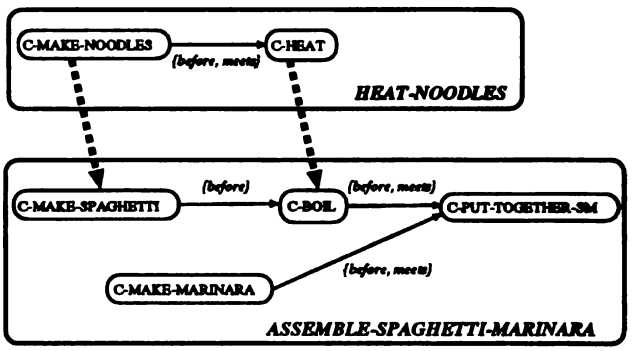


Figure 3: Plan Subsumption

In practice, we cannot expect that all temporal relations will be made explicit in our input. To compare temporal networks properly, we require a constraint propagation procedure such as Allen's [Allen, 1983] to *close* the temporal networks by making explicit any implicit temporal relationships. Allen's constraint propagation algorithm is sound, but unfortunately not complete [Vilain *et al.*, 1989]. This is important because our ability to compare different plans in light of their temporal constraints depends on the extent to which the temporal constraints are made explicit. As a result, we are faced with several standard alternatives: (1) Adopt an approximation algorithm such as Allen's, and live with the possibility that some plan subsumption relationships may remain undetected. There is a family of variations on Allen's algorithm which produce successively better approximations, but only at increasingly exorbitant cost. (2) Use an exact, presumably exponential, algorithm. This may be reasonable for relatively small problems. (3) Restrict the

expressiveness of temporal constraints so that exact solutions can be obtained tractably. For example, Vilain, Kautz and van Beek [Vilain *et al.*, 1989] identified a subset of Allen's interval calculus derived from a point-based representation which admits complete polynomial-time constraint propagation. T-REX currently exercises the first option. We expect that practical experience will educate us as to the best choice.

Structural plan subsumption allows T-REX to automatically classify plan taxonomies strictly according to the semantics of the plans. Figure 4 shows a plan taxonomy constructed by T-REX using the set of plan definitions presented in the Appendix. The root of the plan taxonomy is the trivial plan, PLAN. Taxonomies formed by classifying plan networks, like terminological constraint networks in general, enjoy all the benefits of classification cited in Section 1. Furthermore, as plan libraries grow in size and scope, their organization and maintenance becomes increasingly critical. Search procedures can utilize the definitional placement of plans within a taxonomy for accurate and efficient results. Also, since most present day plan libraries are organized by hand, the clerical demands placed on the plan librarian may become burdensome. Our experience with knowledge engineering shows that when confronted with large quantities of information, the enforced semantics of the terminological approach offers significant advantages [Mays *et al.*, 1991b].

4 COMPUTING SUBSUMPTION

Computing subsumption between the concepts associated with nodes amounts to querying K-Rep. In our case, the concept taxonomy will be constructed in advance, so the results can be retrieved in constant time (we can precompute the transitive closure of the subsumption relations for each concept). Temporal constraints can be represented as bitstrings of length 13, so subsumption between them can also be determined in constant time.

The crux of the plan subsumption problem is to establish a suitable mapping from one plan network to another. This problem is clearly in NP, and there is a polynomial time transformation from directed subgraph isomorphism, which is NP-complete, to subsumption mapping between terminological constraint networks. Thus we have:

Theorem 2 *Subsumption mapping between terminological constraint networks is NP-complete.*

Details of the proof are in [Weida, 1992].

We share the view of Doyle and Patil, who argue against the *restricted language thesis* that KR systems should limit their expressiveness to achieve polynomial worst-case response times [Doyle and Patil, 1991]. It is important to observe that in our instantiation of the subgraph isomorphism problem, both the nodes and the arcs are "labeled," so powerful heuristics can be brought to bear. As Sowa noted, albeit in a different context: "The labels help to guide the pattern match when it is going to be successful, and a mismatch of labels can cause it to fail quickly when there is no chance of success. Therefore, the labels speed up the pattern matching in many cases."

Terminological network subsumption exemplifies the well-known constraint satisfaction problem (CSP). In CSP, we are given a set of variables (corresponding to nodes in the putative subsuming network), and our task is to instantiate each variable with values from a specified domain (nodes in the putative subsumee), subject to certain constraints (for plan networks, the action types of the nodes plus the set of temporal relationships described by the arcs).

We now summarize an algorithm to decide whether some terminological constraint network *T1* subsumes another one *T2*:

1. **Macro Expansion:** Expand each macro node by replacing it with its constituent nodes (recursively). Propagate constraints on a macro node to each of its constituents using a procedure such as Allen's.
2. **Closure:** Close both networks via constraint propagation.
3. **Preliminary Analysis:** For each node *N1* in *T1*, determine which nodes in *T2* are subsumed by *N1* according to the concept taxonomy. Call those nodes the *potential images* of *N1*. If the number of potential images for any node in *T1* is zero, return *false*. Otherwise, sort the nodes of *T1* in increasing order of potential image count to help guide the subsequent graph matching process.
4. **Matching by Backtracking:** Using the preliminary analysis for heuristic guidance, extend the mapping from *T1* to *T2* one step at a time. Each extension consists of selecting an additional node *N1* from *T1* and associating with it an additional node *N2* from among its potential images, such that the constraints on all nodes selected from *T2* continue to respect the constraints on the corresponding nodes from *T1*. When each node from *T1* has been mapped to a distinct node from *T2*, return the mapping. At any point, if there is a node from *T1* which cannot be so mapped, backtrack. If the backtracking process is exhausted without finding a suitable mapping, return *false*.

T-REX currently implements this algorithm, which is sound, and complete to the extent that constraint propagation on T2 is complete. Similar to an existing algorithm for production rule subsumption [Yen *et al.*, 1991], it employs well-known CSP techniques. CSP has been widely studied, and improvements are possi-

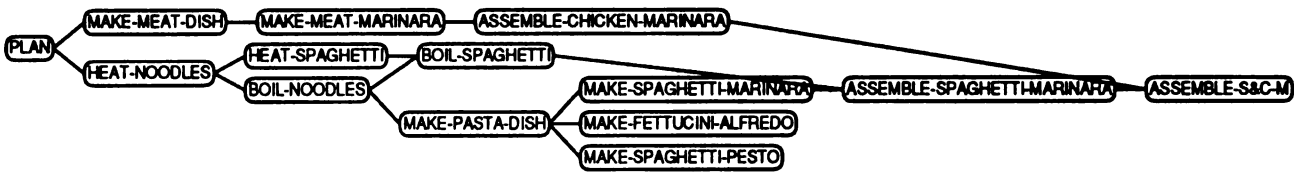


Figure 4: A Definitional Plan Taxonomy

ble. The preliminary analysis that restricts a node's image to be one of its potential subsumees is an example of the *node consistency* technique. Many other powerful CSP methods such as those based on *arc consistency* are available [Mackworth, 1977]. Choosing the optimal mix is domain-dependent and largely still a black art [Kumar, 1992]. For example, we will experiment with the tradeoffs involved in interleaving search to achieve (partial) arc consistency with the backtracking, in using intelligent (e.g., dependency-directed) backtracking, and in exploiting domain-specific techniques.

5 PLAN RECOGNITION

We now exploit the plan library's definitional nature to guide plan recognition. By searching for suitable mappings between the observations and the plans, we can assign the plans modalities, e.g., *necessary*, *possible* and *impossible*, that indicate their status with respect to the observations. This process, which partitions the plan library by modality, is unique to our work. We shall examine plan recognition under varying assumptions about the accuracy and monotonicity of the observations.

An observation represents a determination that actions(s) have occurred and/or that temporal constraint(s) hold between actions. The system records its observations in a network similar to plan networks which we call the *observation network*. Action instances are associated with the nodes of an observation network. In general, the observation network may be an inexact or incomplete model of the events.

As events unfold and observations are made, the observation network is updated, yielding successive versions. An update may entail *extension* and/or *refinement*. Extensions add new actions and/or temporal constraints, while refinements further constrain existing actions and/or temporal constraints. More generally, observations can be retracted or generalized.

We make a *complete library assumption* that each observed action is directed towards fulfilling a plan or plans in the plan library. Consequently, at least one plan is possible at all times and at least one plan will eventually prove necessary. Until Section 5.4, we further make the *single plan assumption* that the observations will ultimately be fully accounted for by a single plan (they may also be partially accounted for by more general plans). Both assumptions are common in the field of plan recognition. The latter is the most restricted version of Kautz's *minimum cardinality assumption*, which always prefers to account for observations with the smallest number of plans [Kautz, 1991]. It is often reasonable to suppose that observed actions are related.

Terminological plan recognition is based on *potential* subsumption relationships between the observations and plans in the plan library. We will say that a plan is *possible* with respect to the observations if it might eventually subsume the observations, i.e., perhaps pending suitable further observations. When a plan cannot subsume the observations under the prevailing assumptions, the plan is *impossible*. A possible plan which actually subsumes the observations is also *necessary*. Stronger plan recognition results may follow from cardinality assumptions, e.g., due to the complete library assumption, we know that when only one plan remains possible, it is effectively necessary. For convenience, we refer to plans which are possible but not necessary as *optional*. Before any observations are made, all plans are optional except for PLAN, which is trivially necessary. Afterwards, both the plan definitions and the prevailing assumptions interact with the observations to determine the modality of each plan.

The recognition process relies on the definitional nature of the plan taxonomy to partition the taxonomy into three connected regions. Figure 5 illustrates the division of the plan taxonomy into necessary (N), optional (O), and impossible (I) regions. Since the taxonomy is definitional, we need not compare every plan with the observations to accomplish the partitioning, e.g., except for PLAN, a plan is not possible unless one of its parents is possible.

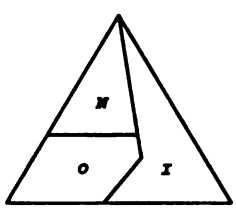


Figure 5: Modalities in a Definitional Plan Taxonomy

5.1 PERFECT OBSERVATIONS

Let us begin with the stringent assumption that the observation network is perfect. In our framework, this implies that the types of observed actions are leaves in the action taxonomy and that observed temporal relationships are nondisjunctive. The observation network may be extended with additional actions, as well as with temporal constraints between the additional actions or between an additional action and a previously observed action. Existing actions and temporal relationships may not be modified or retracted.

The *perfect observation assumption* is sometimes quite justified. For instance, we can flawlessly capture a user's interactions with software systems such as operating systems or graphical user interfaces. Indeed, we view user interfaces as a likely application for our ideas.

Suppose we have the following plan network with two actions (also shown in Figure 3 and in the Appendix):

- HEAT-NOODLES:
  C-MAKE-NOODLES {*before, meets*} C-HEAT

It subsumes the following (unrelated) observation networks, among others:

- OBS-1: MAKE-SPAGHETTI1 {*before*} BOIL2
- OBS-2: MAKE-ZITI3 {*meets*} BAKE4

Thus, both of the preceding observation networks license the conclusion that the HEAT-NOODLES plan is necessary.

Intuitively, a plan is possible with respect to the observations if it subsumes or might eventually subsume them, i.e., it is necessary or optional. We introduce *inverse subsumption* to characterize optional plans which directly reflect the present observations:

Definition 4 *An* inverse subsumption mapping *from terminological constraint network* T2 *to terminological constraint network* T1 *maps every node* N2 *of* T2 *to a distinct node* N1 *of* T1 *such that* N2 *is subsumed by* N1, *and every arc between a pair of nodes in* T2 *is subsumed by the arc between the corresponding nodes in* T1.

An optional plan network P which enjoys an inverse subsumption mapping from the observations will actually subsume them if we subsequently observe nodes and arcs subsumed by the as yet unobserved portion of P. Since an inverse subsumption mapping constitutes direct evidence that a plan may be in progress, we will call such plans *directly optional*. The next observation network, which consists of a single action, is potentially subsumed by HEAT-NOODLES in this way:

- OBS-3: MAKE-SPAGHETTI5

For example, OBS-3 would become subsumed by HEAT-NOODLES if a C-BOIL action were observed to occur after the MAKE-SPAGHETTI5. Hence, the status of HEAT-NOODLES would change from directly optional to necessary. On the other hand, an observation of MAKE-CHICKEN would render HEAT-NOODLES not directly optional, given that a MAKE-CHICKEN action is not subsumed by any action in HEAT-NOODLES.

There is one other class of optional plans, not covered by Definition 4, which may eventually subsume the observations. To see this, consider again OBS-3, now in the context of the portion of Figure 4 detailed in Figure 6 (which shows the pertinent subsumption mapping). The MAKE-SPAGHETTI5 has no counter-

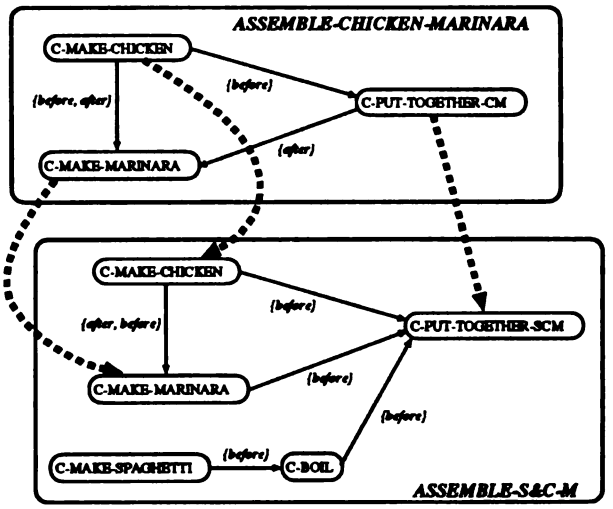


Figure 6: Indirect Optionality

part in the ASSEMBLE-CHICKEN-MARINARA plan, nor have we observed any other action in that plan. A notion of possibility based on inverse subsumption alone would lead to the conclusion that ASSEMBLE-CHICKEN-MARINARA is "impossible." However, it admits the possibility that the agent is following the ASSEMBLE-S&C-M plan. This seems somewhat paradoxical since ASSEMBLE-CHICKEN-MARINARA subsumes ASSEMBLE-S&C-M. Based on our evidence that the latter is optional, we want to sanction the indirect conclusion that ASSEMBLE-CHICKEN-MARINARA is also optional. T-REX therefore recognizes a supplemental class of optional plans. Any plan which does not enjoy an inverse subsumption mapping from the observations, but does subsume an optional plan, is itself *indirectly optional*. Now, we can formally define *potential subsumption*:

Definition 5 *Plan network* P potentially subsumes *observation network* O *under perfect observation iff (1)* P *subsumes* O, *or (2) there exists an inverse subsumption mapping from* O *to* P, *or (3) there exists a plan* P' *such that* P *subsumes* P' *and* P' *potentially subsumes* O.

Potential subsumption expands the notion of actual subsumption, i.e., subsumption entails potential subsumption but the converse is not true. The three cases cover the necessary, directly optional, and indirectly optional plans, respectively. That potential subsumption precisely captures our idea of possibility is stated in the following theorem, proved in [Weida, 1992]:

Theorem 3 *Under the complete library, single plan and perfect observation assumptions, a plan is possible iff it potentially subsumes the observations.*

Given OBS-3, we can now see that our recognition methodology will partition the plan taxonomy in Figure 4 into the following modalities:

Directly Optional:	HEAT-NOODLES
	HEAT-SPAGHETTI
	BOIL-NOODLES
	MAKE-PASTA-DISH
	MAKE-SPAGHETTI-PESTO
	MAKE-SPAGHETTI-MARINARA
	BOIL-SPAGHETTI
	ASSEMBLE-SPAGHETTI-MARINARA
	ASSEMBLE-S&C-M
Indirectly Optional:	MAKE-MEAT-DISH
	MAKE-MEAT-MARINARA
	ASSEMBLE-CHICKEN-MARINARA
Impossible:	MAKE-FETTUCINI-ALFREDO

Note that the recognition of MAKE-FETTUCINI-ALFREDO as impossible depends crucially on the complete library and single plan assumptions. Of course, given a wide range of cooking plans, OBS-3 alone would render many of them impossible.

If observation network OBS-3 is extended to include an instance of C-BOIL occurring after MAKE-SPAGHETTI5, the following plans change from directly optional to necessary: HEAT-NOODLES, HEAT-SPAGHETTI, BOIL-NOODLES and BOIL-SPAGHETTI. By contrast, if OBS-3 is instead extended to include an instance of MAKE-CHICKEN (temporally unconstrained), then HEAT-NOODLES, HEAT-SPAGHETTI, BOIL-NOODLES, MAKE-PASTA-DISH, MAKE-SPAGHETTI-MARINARA, BOIL-SPAGHETTI, and ASSEMBLE-SPAGHETTI-MARINARA change from directly optional to indirectly optional, while MAKE-SPAGHETTI-PESTO changes from directly optional to impossible.

Note that our recognition methodology is not limited to plans. Definition 4 simply specifies that terminological constraint network *T2* satisfies a subnetwork of *T1*. Likewise, Definition 5 applies generally to any pair of terminological constraint networks, *P* and *O*.

5.2 MONOTONIC OBSERVATIONS

Now we permit imprecise observations, including action instances of arbitrarily abstract type and/or disjunctive temporal constraints, along with refinement of prior observations. The type of an action instance in the observation network may be refined to a more specific type [1] Similarly, an observed temporal constraint may be refined to a subset of its disjuncts.

This framework poses more of a challenge. An action instance in the observation network which is not subsumed by a certain action in a plan network may later be refined to the point that it becomes subsumed by that action, and similarly for temporal constraints. For motivation, consider the following pair of plan networks, neither of which subsumes the other (as defined in the Appendix and illustrated in Figure 4):

- BOIL-NOODLES:
  C-MAKE-NOODLES {*before, meets*} C-BOIL
- HEAT-SPAGHETTI:
  C-MAKE-SPAGHETTI {*before*} C-HEAT

Also consider the following pair of observation networks:

- OBS-4: MAKE-NOODLES6 {*before, meets*} BOIL7
- OBS-5: MAKE-SPAGHETTI8 {*before*} HEAT9

Notice that OBS-4, which is subsumed by BOIL-NOODLES, would become subsumed by HEAT-SPAGHETTI if MAKE-NOODLES6 was refined to be of type C-MAKE-SPAGHETTI and the temporal constraint was refined to {*before*}. Conversely, OBS-5, which is subsumed by HEAT-SPAGHETTI, would become subsumed by BOIL-NOODLES if HEAT9 was refined to be of type C-BOIL.

The possibility of refinement forces us to expand the conditions under which a plan is deemed possible. A plan is possible if the observations are consistent with it or may become so. Potential subsumption of the observation network by a plan network under monotonic observation depends on *compatibility* of actions and temporal constraints. We formalize this notion with respect to structural subsumption and our completeness assumptions by the following series of definitions.

Definition 6 *A pair of concepts (constraints) are* compatible *iff there exists a concept (constraint) which they both subsume.*

Thus C-HEAT is compatible with C-BOIL and *vice versa*. Recall that subsumption is reflexive.

Definition 7 *An instance* I *and a generic concept* G *are* compatible *iff the type of* I *is compatible with* G.

Thus HEAT27 is compatible with C-BOIL and conversely.

Definition 8 *Temporal constraints are* compatible *iff the intersection of their disjuncts is non-empty.*

[1] The *type* of an instance is the conjunction of the concepts which subsume it.

The constraint {*before, during*} is bidirectionally compatible with {*during, after*}.

Of course, when an observed action or constraint is incompatible with a certain action or constraint in the plan library, their incompatibility is impervious to future refinement of the observation (recall our assumption that the action taxonomy is complete).

Compatibility for a pair of terminological constraint networks, such as a plan network and an observation network, can be decided as follows:

Definition 9 *A pair of nodes (arcs) are* compatible *iff the associated concepts (constraints) are compatible.*

Definition 10 *There is a* compatibility mapping *from terminological constraint network* T1 *to terminological constraint network* T2 *iff every node of* T1 *is compatible with a distinct node of* T2, *such that every arc between a pair of nodes in* T1 *is compatible with the arc between the corresponding nodes in* T2.

Finally, we can give a more general definition for potential subsumption, which is implemented by T-REX:

Definition 11 *Plan network* P potentially subsumes *observation network* O *under monotonic observation iff (1) there exists a compatibility mapping from* P *to* O, *or (2) there exists a compatibility mapping from* O *to* P, *or (3) there exists a plan* P' *such that* P *subsumes* P' *and* P' *potentially subsumes* O.

Intuitively, a plan network is possible if the observation network can be extended and/or refined so that it is subsumed by the plan network. Definition 11 is formally justified by the following, proved in [Weida, 1992]:

Theorem 4 *Under the complete library, single plan and monotonic observation assumptions, a plan is possible iff it potentially subsumes the observations.*

Returning to our motivating example, Definition 11 shows that BOIL-NOODLES and HEAT-SPAGHETTI potentially subsume observation networks OBS-4 and OBS-5. In particular, the full partitioning of the plan taxonomy in Figure 4 given OBS-4 is:

Necessary:	HEAT-NOODLES
	BOIL-NOODLES
Directly Optional:	HEAT-SPAGHETTI
	MAKE-PASTA-DISH
	MAKE-FETTUCINI-ALFREDO
	MAKE-SPAGHETTI-PESTO
	MAKE-SPAGHETTI-MARINARA
	BOIL-SPAGHETTI
	ASSEMBLE-SPAGHETTI-MARINARA
	ASSEMBLE-S&C-M
Indirectly Optional:	MAKE-MEAT-DISH
	MAKE-MEAT-MARINARA
	ASSEMBLE-CHICKEN-MARINARA

In contrast, the partitioning resulting from OBS-5 is:

Necessary:	HEAT-NOODLES
	HEAT-SPAGHETTI
Directly Optional:	BOIL-NOODLES
	MAKE-PASTA-DISH
	MAKE-SPAGHETTI-PESTO
	MAKE-SPAGHETTI-MARINARA
	BOIL-SPAGHETTI
	ASSEMBLE-SPAGHETTI-MARINARA
	ASSEMBLE-S&C-M
Indirectly Optional:	MAKE-MEAT-DISH
	MAKE-MEAT-MARINARA
	ASSEMBLE-CHICKEN-MARINARA
Impossible:	MAKE-FETTUCINI-ALFREDO

Under the single plan assumption and monotonic observation, the set of plans that are optional (directly or indirectly) decreases monotonically as observations occur. In that case, the effect of each new observation is to change the status of zero or more plans to necessary or impossible. For example, if MAKE-NOODLES6 in OBS-4 is refined to be of type C-MAKE-SPAGHETTI and the temporal constraint refined to {*before*}, the plans HEAT-SPAGHETTI and BOIL-SPAGHETTI change from directly optional to necessary, while MAKE-FETTUCINI-ALFREDO changes from directly optional to impossible. If HEAT9 in OBS-5 is refined to be of type C-BOIL, the plans BOIL-NOODLES and BOIL-SPAGHETTI change from directly optional to necessary.

Notice that under the complete library assumption and monotonic observation the candidate plans will have temporal constraints and constraints on their actions which are the intersection of the constraints in the observations with constraints in the plan library. We intend to enhance T-REX to propagate these new constraints through the observation network, improve the plan library partitioning if it can, and repeat the cycle until arriving at a fixpoint. This requires T-REX to coordinate inferences in K-Rep and MATS.

As before, our recognition methodology for monotonic observations applies to any type of terminological constraint network, not just plans.

5.3 UNRESTRICTED OBSERVATIONS

T-REX actually provides for arbitrary modification and retraction of observations. To reach any useful conclusions, it is necessary to assume in advance that generalization and retraction will not happen. Thus our existing definition of potential subsumption under monotonic observation still applies. When allowing nonmonotonic observations, however, plans considered "necessary" given some observation network may revert to optional status later on. Indeed, seemingly "impossible" plans may later become possible. If an observed action instance is modified, it is automatically reclassified by K-Rep. Nonmonotonic observation could have unfortunate performance conse-

quences. We must effectively be able to undo any constraint propagation in the observation network, since the justification may cease to exist. Retraction in the observation network is currently done by recomputation. Presumably it could also be supported via truth maintenance, but the cost of tracking dependencies may not be worthwhile.

5.4 SIMULTANEOUS PLANS

When the single plan assumption is violated, T-REX accounts for the eventuality that more than one plan is underway. First, it must be able to relate the observations to a group of plans. T-REX (conceptually) places the nodes from several plans into one plan network, preserving the original constraints on those nodes. Relationships between nodes taken from different plans are unconstrained. Thus, a *multiple plan network* allows its constituent plans to be interleaved in any way.

A set of plans accounts for all observed actions iff there is a compatibility mapping from the observation network to their multiple plan network. T-REX also needs a way to explore the set of possible plan combinations. Kautz's minimum cardinality assumption addresses this problem. His implementation simply considers plans pairwise when a single plan does not suffice to explain the observations, and failing that, three at a time and so on, *ad infinitum* [Kautz, 1991]. As a first cut at improvement, T-REX only considers those multiple plan networks that have a compatible action for every observed action.

As an example, consider observation network OBS-6:

- OBS-6:
  MAKE-FETTUCINI10 {*before*} MAKE-NOODLES11
  {*before*} MAKE-ALFREDO12 {*before*}
  MAKE-ALFREDO13

Since no single plan can account for the observations, T-REX infers that three possible sets of two (interleaved) plans can account for OBS-6:

1. {MAKE-FETTUCINI-ALFREDO,
   MAKE-FETTUCINI-ALFREDO}
2. {MAKE-FETTUCINI-ALFREDO, MAKE-PASTA-DISH}
3. {MAKE-PASTA-DISH, MAKE-PASTA-DISH}

6 N-DIMENSIONAL SPATIAL CONSTRAINT NETWORKS

The constraint satisfaction problem characterizes many important problems in AI and computer science at large [Kumar, 1992]. Often it is formulated in terms of constraint networks. Our methods apply whenever it is useful to reason about structural subsumption between constraint networks or to recognize partial instances of constraint networks via potential subsumption. We now sketch an application to descriptions of spatial configurations.

Disjunctions of Allen's 13 primitives capture all possible relationships between intervals along a single dimension. While Allen's scheme was designed for the temporal domain, it is equally appropriate for one-dimensional space. Moreover, as pointed out in [Mukerjee and Joe, 1990], arbitrary relationships in N-dimensional space can be modeled by n-tuples of Allen's constraints. As a first approximation to spatial relationships, we associate objects and locations with rectilinear bounding boxes aligned to the axes, i.e., we consider the projections onto the axes as intervals, and then use Allen's relations on them.

The alignment can in fact be varied [Mukerjee and Joe, 1990]. The following constraint network specifies a C-SQUARE whose bounding box is disjoint from that of a C-RECTANGLE in 2-dimensional space:

- C-SQUARE ({*before, after*}, {*before, after*}) C-RECTANGLE

Orthogonal constraint networks maintain relationships along each axis. Constraint propagation can be applied independently in each dimension to discover, for example, that if there is an object which is properly contained in the C-SQUARE then it is spatially disjoint from the C-RECTANGLE.

Our idea of constraint network subsumption extends to multiple dimensions: constraint *C1* subsumes constraint *C2* iff each component of *C1* subsumes the corresponding component of *C2* as defined previously. Thus, the preceding description subsumes the following one, which says that a C-SQUARE is left of and above a C-RECTANGLE (assuming normal interpretation of the x and y axes, respectively):

- C-SQUARE ({*before*}, {*after*}) C-RECTANGLE

Based on subsumption, we can automatically classify a library of such spatial descriptions.

Our formulation of potential subsumption also extends directly to multiple dimensions. Spatial subsumption and potential subsumption may be useful for computer vision and graphics tasks. Potential subsumption can recognize spatial configurations of objects described by library entries from partial observations recorded in orthogonal observation networks.

7 CURRENT DIRECTIONS

We are currently focusing on extending the expressive power of our representation and developing associated algorithms for plan subsumption and recognition. For example, a plan may have *equality constraints* between roles of its action concepts to ensure, e.g., that the spaghetti which is formed in one step is the *same*

spaghetti later boiled. Equality constraints in plans resemble *role value maps* in standard TKR [Brachman and Schmolze, 1985]. We would also like to take advantage of MATS' facility for reasoning with metric (as well as qualitative) temporal information. In addition, we are exploring the inclusion of state descriptions, e.g., preconditions and effects of plans and their constituent actions in plan networks, along with states in observation networks. We will use this information to perform plan recognition which includes chaining on preconditions and goals of actions and plans. Finally, as a concrete application for our work, we are considering use of T-REX to enhance the capabilities of the FAME expert system [Apte *et al.*, 1992]. FAME's problem solving components and user interface are all constructed on top of, and integrated through K-Rep. FAME is particularly hospitable to T-REX since all of FAME's input and output is already done via presentation and acceptance of K-Rep concepts.

8 RELATED WORK

Our work in plan subsumption draws upon TKR [Brachman and Schmolze, 1985], as well as Allen's temporal logic [Allen, 1983]. Schmiedel [Schmiedel, 1990] has described an ambitious attempt to extend terminological logic with temporal semantics by integrating both Allen's temporal logic and Shoham's [Shoham, 1987]. Although he offers no algorithm, Schmiedel does suggest a few "preliminary hints" (his words), including a definition of subsumption which corresponds to ours. His work did not consider temporal constraint networks as first class entities, nor did he address either recognition or the notion of potential subsumption. In addition, an important development in reasoning with compositions of terminological concepts is [Yen *et al.*, 1991], which integrates TKR with a production system.

Previous work on plan subsumption allowed plans that were either atemporal and used for plan synthesis [Wellman, 1990] or restricted to the relationship of temporal sequence and used for information retrieval [Devanbu and Litman, 1991]. T-REX's plan subsumption is similar in spirit to CLASP [Devanbu and Litman, 1991], which described plans as action sequences by means of regular expressions. However, by using Allen's temporal logic, T-REX supports simultaneous actions. T-REX also captures finer sequential relations than CLASP, which, for example, makes no distinction between *before* and *meets*. Finally, T-REX plan networks can be composed nicely from binary constraints, making for a compact and facile notation. Regular expressions are comparatively unwieldy monolithic structures. On the other hand, CLASP supports disjunction and looping. Besides T-REX, there are two projects currently extending TKR to handle plans. RAT [Heinsohn *et al.*, 1992] focuses on the representation of complex state descriptions, while PROTODL [Borgida, 1992] reconstructs CLASP using natural semantics.

While there have been many approaches to plan recognition, e.g., [Allen and Perrault, 1980, Carberry, 1990, Cohen and Levesque, 1990, Litman and Allen, 1987, Pollack, 1990, Sidner, 1985], our work is most closely related to that of Kautz [Kautz, 1991]. Our plan recognition technique, like Kautz's, is deductive, and incorporates the use of a plan abstraction taxonomy (as well as the traditional hierarchy decomposing plans into constituent actions). Both approaches are also restricted compared to other techniques in that they do not chain on state information (e.g. preconditions and effects), and have strong assumptions such as plan library correctness and completeness. Kautz gave a formal theory of plan recognition, along with more practical algorithms that approximate his theory. A major contribution was his logical characterization of the completeness assumptions. Unlike Kautz's approach, we extend work in TKR to formalize and automate the organization of the plan taxonomy. Moreover, we directly exploit the library's definitional nature to guide plan recognition. We also use an underlying TKR system, K-Rep, to represent and reason with the actions and objects that are the building blocks of plans, whereas atomic actions and objects in [Kautz, 1991] and many other approaches lack defined semantics. Thus our approach allows the plan recognition system to share the advantages of existing terminological ontologies. Finally, Kautz computes the possible consequences of each observed action independently and records them in separate graph structures which are combined by repeated graph-merging operations. We determine possible consequences from the observation network as a whole, on a context-dependent basis.

9 CONCLUSION

We extend the scope of TKR by showing how to compute structural subsumption relationships among constraint networks, such as temporal networks used in plan representation. In the case of plans, we use K-Rep to define subsumption on structured action concepts and we also define subsumption on temporal constraints. This allows us to automatically organize a plan library into a strict taxonomy, thereby easing search and maintenance tasks. We further exploit the plan library's terminological nature in a new and promising approach to plan recognition that partitions the plan library by modality. Our framework supports arbitrary revision of prior observations. We have explored our ideas in T-REX, a system whose modular architecture utilizes state of the art components: K-Rep for standard TKR and MATS for temporal reasoning. Our ideas apply to constraint networks in general, and we have proposed a representation, subsumption and recognition facility for configurations of objects in N-dimensional space.

Acknowledgements

We thank Alex Borgida, Bob Dionne, Michael Elhadad, Henry Kautz, Eric Mays, Jacques Robin and Sal Stolfo for valuable comments on earlier drafts.

APPENDIX

The following definitions were used by T-Rex to construct the plan taxonomy in Figure 4. The function "defplan" takes three arguments. Informally, the first argument names the plan type being defined. The second argument is a list of plan steps. Each step specifies a label for the step and a constraint on its action type. Any action instance satisfying the constraint must be subsumed by the action type. The third argument is a list of temporal constraints. Each constraint specifies a disjunction of Allen's 13 temporal relationships between the temporal intervals associated with the two designated plan steps. For example, the first definition states that any instantiation of the plan HEAT-NOODLES satisfies the following constraints: it contains an action instance of type C-MAKE-NOODLES; it contains an action instance of type C-HEAT; the temporal interval associated with the first action instance is *before* or *meets* the temporal interval associated with the second. Also note the use of the plan BOIL-SPAGHETTI as a macro action in the definition of ASSEMBLE-SPAGHETTI-MARINARA.

```
(defplan HEAT-NOODLES
   ((s1 C-MAKE-NOODLES)
    (s2 C-HEAT))
   ((s1 (before meets) s2)))

(defplan BOIL-NOODLES
   ((s1 C-MAKE-NOODLES)
    (s2 C-BOIL))
   ((s1 (before meets) s2)))

(defplan HEAT-SPAGHETTI
   ((s1 C-MAKE-SPAGHETTI)
    (s2 C-HEAT))
   ((s1 before s2)))

(defplan BOIL-SPAGHETTI
   ((s1 C-MAKE-SPAGHETTI)
    (s2 C-BOIL))
   ((s1 before s2)))

(defplan MAKE-PASTA-DISH
   ((s1 C-MAKE-NOODLES)
    (s2 C-BOIL)
    (s3 C-MAKE-SAUCE))
   ((s1 (before meets) s2)))

(defplan MAKE-SPAGHETTI-MARINARA
   ((s1 C-MAKE-SPAGHETTI)
    (s2 C-BOIL)
    (s3 C-MAKE-MARINARA))
   ((s1 (before meets) s2)))

(defplan ASSEMBLE-SPAGHETTI-MARINARA
   ((bs BOIL-SPAGHETTI)
    (s3 C-MAKE-MARINARA)
    (s4 C-PUT-TOGETHER-SM))
   (((s2 bs) (before meets) s4)
    (s3 (before meets) s4)))

(defplan MAKE-SPAGHETTI-PESTO
   ((s1 C-MAKE-SPAGHETTI)
    (s2 C-MAKE-PESTO)
    (s3 C-BOIL))
   ((s1 (before meets) s3)))

(defplan MAKE-FETTUCINI-ALFREDO
   ((s1 C-MAKE-FETTUCINI)
    (s2 C-MAKE-ALFREDO)
    (s3 C-BOIL))
   ((s1 (before meets) s3)))

(defplan MAKE-MEAT-DISH
   ((s1 C-MAKE-MEAT)
    (s2 C-MAKE-SAUCE)))

(defplan MAKE-MEAT-MARINARA
   ((s1 C-MAKE-MEAT)
    (s5 C-MAKE-MARINARA)))

(defplan ASSEMBLE-CHICKEN-MARINARA
   ((s1 C-MAKE-CHICKEN)
    (s2 C-MAKE-MARINARA)
    (s3 C-PUT-TOGETHER-CM))
   ((s1 (before after) s2)
    (s1 before s3)
    (s3 after s2)))

(defplan ASSEMBLE-S&C-M
   ((ms C-MAKE-SPAGHETTI)
    (b C-BOIL)
    (mc C-MAKE-CHICKEN)
    (mm C-MAKE-MARINARA)
    (pt C-PUT-TOGETHER-SCM))
   ((ms before b)
    (b before pt)
    (mc before pt)
    (mm before pt)
    (mc (after before) mm)))
```

References

[Allen and Perrault, 1980] J. F. Allen and C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178, 1980.

[Allen, 1983] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[Apte *et al.*, 1992] C. Apte, R. Dionne, J. Griesmer, M. Karnaugh, J. Kastner, M. Laker, and E. Mays. An experiment in constructing an open expert system using a knowledge substrate. IBM Journal of Research and Development, 1992. To appear.

[Borgida *et al.*, 1989] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. Classic: A structural data model for objects. In *Proc. 1989 ACM SIGMOD International Conference on the Management of Data*, pages 58–67, Portland, OR, 1989.

[Borgida, 1992] A. Borgida. Towards the systematic development of terminological reasoners: Clasp reconstructed. In *Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, Cambridge, MA, 1992.

[Brachman and Schmolze, 1985] R. J. Brachman and J. G. Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[Carberry, 1990] S. Carberry. *Plan Recognition in Natural Language Dialogue*. MIT Press, Cambridge, MA, 1990.

[Cohen and Levesque, 1990] P. R. Cohen and H. J. Levesque. Rational interaction as the basis for communication. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 221–255. MIT Press, Cambridge, MA, 1990.

[Devanbu and Litman, 1991] P. T. Devanbu and D. J. Litman. Plan-based terminological reasoning. In *Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 128–138, Cambridge, MA, 1991.

[Doyle and Patil, 1991] J. Doyle and R. Patil. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48(3):261–297, April 1991.

[Heinsohn *et al.*, 1992] J. Heinsohn, D. Kudenko, B. Nebel, and H. J. Profitlich. Rat: Representation of actions using terminological logics. DFKI, 1992.

[Kautz and Ladkin, 1991] H. A. Kautz and P. B. Ladkin. Integrating metric and qualitative temporal reasoning. In *Proceedings of AAAI-91*, pages 241–246, Anaheim, CA, 1991.

[Kautz, 1991] H. A. Kautz. A formal theory of plan recognition and its implementation. In J. Allen, H. Kautz, R. Pelavin, and J. Tenenberg, editors, *Reasoning About Plans*, pages 69–125. Morgan Kaufmann, San Mateo, CA, 1991.

[Kumar, 1992] V. Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, 13(1):32–41, 1992.

[Litman and Allen, 1987] D. J. Litman and J. F. Allen. A plan recognition model for subdialogues in conversation. *Cognitive Science*, 11:163–200, 1987.

[MacGregor and Bates, 1987] R. MacGregor and R. Bates. The loom knowledge representation language. Technical Report ISI/RS-87-188, USC/Information Sciences Institute, Marina del Ray, CA, 1987.

[MacGregor, 1991] R. MacGregor. The evolving technology of classification-based knowledge representation systems. In J. Sowa, editor, *Principles of Semantic Networks*, pages 385–400. Morgan Kaufmann, Los Altos, CA, 1991.

[Mackworth, 1977] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[Mays *et al.*, 1991a] E. Mays, R. Dionne, and R. Weida. K-rep system overview. *Sigart Bulletin*, 2(3):93–97, June 1991. Special Issue on Implemented Knowledge Representation and Reasoning Systems.

[Mays *et al.*, 1991b] E. Mays, S. Lanka, B. Dionne, and R. Weida. A persistent store for large shared knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 3(1):33–41, March 1991.

[Mukerjee and Joe, 1990] A. Mukerjee and G. Joe. A qualitative model for space. In *Proceedings of AAAI-90*, pages 721–727, Boston, MA, 1990.

[Pollack, 1990] M. E. Pollack. Plans as complex mental attitudes. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 77–104. MIT Press, Cambridge, 1990.

[Schmiedel, 1990] A. Schmiedel. A temporal terminological logic. In *Proceedings of AAAI-90*, pages 640–645, Boston, MA, 1990.

[Shoham, 1987] Y. Shoham. Temporal logics in ai: Semantical and ontological considerations. *Artificial Intelligence*, 33(1):89–104, 1987.

[Sidner, 1985] C. L. Sidner. Plan parsing for intended response recognition in discourse. *Computational Intelligence*, 1(1):1–10, Feb 1985.

[Vilain *et al.*, 1989] M. Vilain, H. Kautz, and P. van Beek. Constraint propagation algorithms for temporal reasoning: a revised report. In J. deKleer and D. Weld, editors, *Readings in Qualitative Reasoning About Physical Systems*. Morgan Kaufmann, Los Altos, CA, 1989.

[von Luck *et al.*, 1987] K. von Luck, B. Nebel, C. Pelatson, and A. Schmiedel. The anatomy of the back system. Technical Report KIT- Report 41, Technische Universitat Berlin, Berlin, 1987.

[Weida, 1992] R. Weida. Forthcoming dissertation proposal, 1992.

[Wellman, 1990] M. P. Wellman. *Formulation of Tradeoffs in Planning Under Uncertainty*. Morgan Kaufmann, San Mateo, CA, 1990.

[Woods, 1986] W. A. Woods. Important issues in knowledge representation. *Proceedings of the IEEE*, 74(10):1322–1334, 1986.

[Yen *et al.*, 1991] J. Yen, R. Neches, and R. MacGregor. Clasp: Integrating term subsumption systems and production systems. *IEEE Transactions on Knowledge and Data Engineering*, 3(1):25–31, March 1991.

A Preference Semantics for Defaults in Terminological Logics

J. Joachim Quantz
Technische Universität Berlin
Project KIT-BACK, FR 5–12
Franklinstr. 28/29, D-1000 Berlin 10

Véronique Royer
Onera–Cert
2 av. Edouard Belin-BP 4025
F-31055 Toulouse

Abstract

We specify a model-theoretic semantics for defaults in terminological logics using Shoham's framework of preference semantics. The semantics contains specificity as a precedence criteria between defaults and yields a disjunctive skepticism in case of non-resolvable conflicts. The primary aim of the semantics is the default entailment of descriptions. We indicate, however, several possibilities of an extension towards default subsumption. To establish a connection between the semantics and the actual computation of inferences we introduce the syntactic notions of default spaces and conflicts. We show that they provide a sound though not complete syntactic characterization of default entailment. We also sketch the integration of an additional precedence criterion, namely anteriority, and give a fixed-point characterization of preferential entailment.

1 INTRODUCTION

In this paper we specify a preference semantics for defaults in terminological logics (TL). Our motivation for doing so is our aim to integrate defaults into the terminological representation system BACK. Consequently, our investigations are not purely theory driven and we will leave aside some issues that might be interesting from the viewpoint of nonmonotonic logics but seem less relevant in the context of TL-systems.

Terminological logics typically distinguish between *definitions*, *descriptions*, and *rules*. A definition has the form $t_n \doteq t$ and expresses that the name t_n is used as an abbreviation for the term t. There are two types of terms in terminological logics, namely *concepts* (unary predicates) and *roles* (binary predicates). In a description an object is described as being an instance of a concept ($o :: c$). Rules have the form $c_1 \rightarrow c_2$ and stipulate that each instance of the concept c_1 is also an instance of the concept c_2. Note that from a theoretical point of view definitions and rules can both be reduced to terminological formulas $t_1 \sqsubseteq t_2$ saying that t_2 *subsumes* t_1. (Thus, $t_n \doteq t$ can be rewritten as $t \sqsubseteq t_n \wedge t_n \sqsubseteq t$.) In the following we use α for descriptions, θ for subsumptions, and γ for arbitrary TL-formulas.

The terminological logic of BACK V5 is specified by the following syntax (t_p are primitive terms, t_n term names):

$$\begin{array}{lcl} c & \rightarrow & \top, \bot, c_p, c_n, \neg c_p, c_1 \sqcap c_2, r{:}o, \\ & & \forall\, r{:}c_1, \geq n\, r{:}c_1, \leq n\, r{:}c_1 \\ r & \rightarrow & \bot, r_p, r_n, \neg r_p, r_1 \sqcap r_2, \\ & & r_1^{-}, r_1.r_2, c|r_1, r_1|_c \\ \gamma & \rightarrow & t_1 \sqsubseteq t_2, o :: c \end{array}$$

Note that the fact that two object o_1 and o_2 stand in the relation r, i.e. that o_2 is a role-filler for r at o_1 is expressed as $o_1 :: r{:}o_2$.

We assume the usual modeltheoretic semantics where a model M of a set of TL-formulas Γ is a pair $\langle D, \mathcal{I} \rangle$. The interpretation function $[\cdot]^{\mathcal{I}}$ maps concepts into subsets of the domain D, roles into subsets of $D \times D$, and object-names injectively into D, respecting the following equations (we use $r(d)$ to denote $\{e : \langle d, e \rangle \in r\}$):

$$[\top]^{\mathcal{I}} = D \quad (1)$$

$$[\bot]^{\mathcal{I}} = \emptyset \quad (2)$$

$$[\neg t_p]^{\mathcal{I}} = D \setminus [t_p]^{\mathcal{I}} \quad (3)$$

$$[t_1 \sqcap t_2]^{\mathcal{I}} = [t_1]^{\mathcal{I}} \cap [t_2]^{\mathcal{I}} \quad (4)$$

$$[\forall r : c]^{\mathcal{I}} = \{d : [r]^{\mathcal{I}}(d) \subseteq [c]^{\mathcal{I}}\} \quad (5)$$

$$[\geq nr : c]^{\mathcal{I}} = \{d : |[r]^{\mathcal{I}}(d) \cap [c]^{\mathcal{I}}| \geq n\} \quad (6)$$

$$[\leq nr : c]^{\mathcal{I}} = \{d : |[r]^{\mathcal{I}}(d) \cap [c]^{\mathcal{I}}| \leq n\} \quad (7)$$

$$[r : o]^{\mathcal{I}} = \{d : [o]^{\mathcal{I}} \in [r]^{\mathcal{I}}(d)\} \quad (8)$$

$$[r_1^{-}]^{\mathcal{I}} = \{\langle d, e \rangle : \langle e, d \rangle \in [r]^{\mathcal{I}}\} \quad (9)$$

$$[r_1.r_2]^{\mathcal{I}} = [r_1]^{\mathcal{I}} \circ [r_2]^{\mathcal{I}} \quad (10)$$

$$[c|r]^{\mathcal{I}} = [r]^{\mathcal{I}} \cap ([c]^{\mathcal{I}} \times D) \quad (11)$$

$$[r|_c]^{\mathcal{I}} = [r]^{\mathcal{I}} \cap (D \times [c]^{\mathcal{I}}) \quad (12)$$

Satisfaction of formulas is then defined as follows:

$$M \models t_2 \sqsubseteq t_1 \quad \text{iff} \quad [t_2]^{\mathcal{I}} \subseteq [t_1]^{\mathcal{I}} \quad (13)$$

$$M \models o :: c \quad \text{iff} \quad [o]^{\mathcal{I}} \in [c]^{\mathcal{I}} \quad (14)$$

A structure M is a model of a formula γ iff $M \models \gamma$; it is a model of a set of formulas Γ iff it is a model of every formula in Γ. A formula γ is *entailed* by a set of formulas Γ (written $\Gamma \models \gamma$) iff every structure which is a model of Γ is also a model of γ. A set of TL-formulas Γ is *satisfiable* iff it has a model, otherwise it is *inconsistent*.

An application is modeled in TL by a list of definitions, rules, and descriptions, i.e. a set of TL-formulas Γ. On the basis of such a modeling, the following types of queries are meaningful:

- Is a term t_1 more special than a term t_2, i.e., is t_1 *subsumed* by t_2?
  $\Gamma \models t_1 \sqsubseteq t_2$
- Is an object o an instance of concept c (object classification)?
  $\Gamma \models o :: c$
- Are two objects o_1,o_2 related by a role r, i.e., is o_2 a role-filler for r at o_1?
  $\Gamma \models o_1 :: r{:}o_2$
- Which objects are instances of a concept c (retrieval)?
- Is a description α inconsistent with the modeling (consistency check)?
  Is $\Gamma \cup \{\alpha\}$ inconsistent?

Note that, from a theoretical point of view, these queries are all reducible to entailment of subsumptions or descriptions. Furthermore, entailment of descriptions can be reduced to entailment of subsumption. From an implementation oriented perspective, however, special algorithms are needed for processing different types of queries in order to guarantee efficient performance (confer, for example, [Kindermann 90] for the issue of retrieval).

Definitions, descriptions, and rules can only be used to model strict knowledge about a domain. In most applications, however, arises the need to model rules that are not strict but allow for exceptions. It is exactly this kind of knowledge that we think should be modeled by defaults. We will therefore integrate defaults of the form $c_1 \rightsquigarrow c_2$ as a fourth kind of knowledge into TL.[1] We can paraphrase such a default as "whenever an object is an instance of c_1 it is also an instance of c_2 unless this is in conflict with other knowledge". (A conflict occurs when an object is an instance of two disjoint concepts.) We thus view defaults as a means to derive additional information for an object from the strict information available. Consequently, we will focus on inferences on the object level, whereas much literature has stressed default consequences on the terminological level (e.g., [Padgham 89]).

Note that, because of their rule-format, defaults "inherit" all the problems inherent in rules, in particular the *problem of disjunction* and the *lifting problem*. We will deal with these issues in detail in Section 3.

[1]This format of defaults in terminological logics was first proposed by Pfahringer in [Pfahringer 89].

Our conception of defaults has rather drastic consequences for the status of conflicts in TR-systems. In the current BACK-system conflicting information is simply rejected. Since all knowledge is regarded as strict the conflict can only arise from incorrect modeling: either the definition of a term or a rule is not correct, or the object description itself is wrong. But if by defaults an object is an instance of two disjoint concepts we just have to assume that it is an exception to one of the defaults. Since we cannot reject conflicting information in this case we have to find a way to resolve the conflict. Due to this new status of conflicts we see three major tasks involved in the integration of defaults into terminological logics: the *semantic task* consists in specifying the meaning of defaults: when is a TL-formula entailed by a set of TL-formulas Γ and a set of defaults Δ. In doing so we have to decide how to handle conflicting defaults, i.e., we have to check whether our semantics is compatible with our intuition about default behavior.

Defining the set of entailed formulas is one thing, computing it is something else. The computation of consequences induced by strict rules is straightforward: derivable information can be collected incrementally since entailment is monotonic. As a consequence of the non-monotonicity of defaults, determining which information is really derivable is non-trivial—either all applicable defaults are determined in advance or derived information has to be revised in the course of computation. We call this task the *computation task*.

Related to the computation task is the *revision task*. Whereas the computation task is concerned with deriving entailed formulas for a given Γ and Δ, the revision task has to deal with incrementally extending Γ. This task arises since TL-systems usually cache information derived from Γ and support incremental modeling of descriptions.

In this paper we will focus on the semantic task. We will follow Shoham's approach and specify a preference semantics. In Sections 4 and 5 we will then provide a sound though incomplete syntactic characterization of our semantics which serves as a basis for the computation task. We will not say anything about the revision task, however.[2]

2 THE PREFERENCE SEMANTICS

Shoham presents a general framework for non-monotonic logics: define a preference relation $\sqsubset$ on the models of a logic; then a model M_1 is preferred iff there is no model M_2 such that $M_1 \sqsubset M_2$ and a set of formulas Γ preferentially entails γ (written $\Gamma \models_{\sqsubset} \gamma$) iff all preferred models of Γ are models of γ [Shoham 88].

In the following we will thus specify a preference relation on TL-models. Before we can do so, however, we have to decide how to handle conflicting defaults. We think that the

[2]The revision task is related to the general problem of revising descriptions. Kindermann has investigated this problem in [Kindermann 92].

framework of terminological logics provides *specificity* as a straightforward criterion for conflict resolution. In order to define this notion we start with a definition of defaults:

Definition 1 *A default δ has the form $c_1 \rightsquigarrow c_2$ where c_1 and c_2 are arbitrary concepts. We say that c_1 is the* **premise** *of δ (written δ_p) and that c_2 is the* **conclusion** *of δ (written δ_c).*

Now we can define a precedence relation on defaults reflecting their specificity with respect to the hierarchy resulting from the terminology and the rules:

Definition 2 *Let Γ be any set of TL-formulas and δ_1, δ_2 any defaults. We say that δ_1* **precedes** *δ_2 wrt Γ (written $\delta_1 \prec_\Gamma \delta_2$) iff $\Gamma \models \delta_{1p} \sqsubseteq \delta_{2p} \wedge \Gamma \not\models \delta_{2p} \sqsubseteq \delta_{1p}$.*

Now our resolution strategy will be roughly the following: a default δ is cancelled if it conflicts with a default δ' which precedes it.

Note that the precedence relation is a partial order on defaults which can be computed from a given set of definitions, rules, and defaults by using standard subsumption algorithms. Note further that additional precedences between defaults might be specified by the user to refine this ordering. As long as the resulting precedence relation remains a partial ordering this extension can easily be integrated into the general approach presented here.[3]

Now that our intuition has been stated we can begin to specify the preference relation on models which is supposed to express our intuition in formal terms. The basic idea is to prefer models with as few exceptions as possible. Note that we do not compare arbitrary models, however. Two models will be comparable only if they have the same domain D. Thus exceptions refer to the individuals of the underlying domain. Furthermore, we assume the usual "Unique Name Assumption", reflected in the stipulation that interpretation functions $\mathcal{I}$ map object-names *injectively* into D. We thus compare what Bossu and Siegel call *discriminant models* [Bossu, Siegel 85].

We now define exceptions and fulfillers of defaults:

Definition 3 *Let δ be any default and M any structure with domain D.*
The set of **exceptions** *to δ in M is defined as*

$$E_M(\delta) \stackrel{\text{def}}{=} \{d \in D : d \in [\![\delta_p]\!]^{\mathcal{I}} \wedge d \notin [\![\delta_c]\!]^{\mathcal{I}}\}.$$

The set of **fulfillers** *for δ in M is defined as*

$$F_M(\delta) \stackrel{\text{def}}{=} \{d \in D : d \in [\![\delta_p]\!]^{\mathcal{I}} \wedge d \in [\![\delta_c]\!]^{\mathcal{I}}\}.$$

Since we will compare two models with respect to their exceptions it is useful to define two sets: one set containing those entities which are exceptions to a default in one model but not in the other; and one set containing those entities which fulfill a default in one model but not in the other. Since it is also important to which defaults the entities are exceptions we will rather focus on sets containing entity-default-pairs.

[3] Additional precedences will impact the redundancy of defaults, however (cf. Section 3.3).

Definition 4 *Let M_1 and M_2 be any models with common domain D. The set of* **malus pairs** *of M_1 with respect to M_2 is defined as*

$$E_2^1 \stackrel{\text{def}}{=} \{\langle \delta, d\rangle : d \in E_{M_1}(\delta) \setminus E_{M_2}(\delta)\}.$$

The set of **bonus pairs** *of M_1 with respect to M_2 is defined as*

$$F_2^1 \stackrel{\text{def}}{=} \{\langle \delta, d\rangle : d \in F_{M_1}(\delta) \setminus F_{M_2}(\delta)\}.$$

We can then define exception-preference as our first preference criterion between models.

Definition 5 *Let Γ be any set of TL-formulas, M_1 and M_2 any models of Γ, and Δ any set of Defaults. M_2 is* **exception-preferred** *to M_1 wrt Γ and Δ (written $M_1 \sqsubset_E M_2$) iff*

1. *M_1 and M_2 are discriminant models over the same domain D,*
2. *$E_1^2 \neq E_2^1$,*
3. *$\forall \langle \delta, d\rangle \in E_1^2 : \exists \langle \delta', d'\rangle \in E_2^1 : \delta' \prec_\Gamma \delta$.*

In other words, if M_2 contains an exception which is no exception in M_1 this exception must be justified by being an exception at a preceded default.

Note that exception-preference captures the classical minimization principle of Circumscription, namely the minimization of the extensions of *abnormality* predicates [McCarthy 86, Gelfond et al. 89]. Our exception-preference indeed minimizes the exception sets.

Proposition 1 *Let Γ be any set of TL-formulas, M_1 and M_2 any discriminant models of Γ over the same domain D, and Δ any set of Defaults. If E_1^2 is empty but E_2^1 is not, then $M_1 \sqsubset_E M_2$.*

Proof: Condition 1 and 2 in definition 5 are satisfied by assumption, condition 3 is trivially true since E_1^2 is empty. □

Note further that exception-preference gives rise to a partial pre-ordering over the set of finite models. It corresponds to an aggregation of the precedence relation between exceptional defaults. This kind of aggregation is called *elitism aggregation* in [Cayrol et al. 92] and proved to yield a partial pre-ordering provided that the underlying precedence relation is bounded, i.e. has no infinite strictly increasing chains. This is clearly the case in the context of TL if we consider finitely many defaults and finite domains.

We will illustrate the effect of exception-preference by an example. We use the following notational conventions in our example: we only list the "relevant" aspects of the "relevant" models. Thus, if a model has the entry c ($\bar{c}$) in the column of an object-name o this means that the interpretation of o is (not) an element of the interpretation of c in this model. We then list the exceptions and sometimes the fulfillers in the model: the entry in the column E_i indicates the exceptions to default δ_i in the model. We mark preferred models with $*$ and occasionally use $\circ$ to indicate models

		o	o	E_1	E_2	E_3	F_1	F_2	F_3
γ_1: o :: $c_1 \sqcap c_3$	$*M_1$:	c_2	c_4	$\emptyset$	$\emptyset$	$\{o\}$	$\{o\}$	$\{o\}$	$\emptyset$
δ_1: $c_1 \rightsquigarrow c_2$	$*M_2$:	c_2	$\bar{c}_4$	$\emptyset$	$\{o\}$	$\emptyset$	$\{o\}$	$\emptyset$	$\{o\}$
δ_2: $c_2 \rightsquigarrow c_4$	M_3:	$\bar{c}_2$	c_4	$\{o\}$	$\emptyset$	$\{o\}$	$\emptyset$	$\emptyset$	$\emptyset$
δ_3: $c_3 \rightsquigarrow \neg c_4$	oM_4:	$\bar{c}_2$	$\bar{c}_4$	$\{o\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{o\}$
$M_3 \sqsubset_E M_1, M_4 \sqsubset_F M_2$									
$\models_\Delta o :: c_2, \not\models_\Delta o :: c_4, \not\models_\Delta o :: \neg c_4$									

Figure 1: Fulfillers are Needed to Cope with Default-Chains.

which would be preferred if our definitions were weaker. The following example illustrates exception-preference and the effect of specificity:

		o	E_1	E_2
γ_1: $c_2 \sqsubseteq c_1$	M_1:	c_3	$\emptyset$	$\{o\}$
γ_2: o :: c_2	$*M_2$:	$\bar{c}_3$	$\{o\}$	$\emptyset$
δ_1: $c_1 \rightsquigarrow c_3$				
δ_2: $c_2 \rightsquigarrow \neg c_3$				
$M_1 \sqsubset_E M_2$ (because of specificity)				
$\models_\Delta o :: \neg Cc$				

The exception criterion allows to prefer models with as few (and as least specific) exceptions as possible. However, this criterion is not sufficient to capture the intuitive view that as much defaults as possible must be fired.[4] Consequently, we need an additional criterion, called *fulfillment criterion*, for maximizing the fulfiller sets.

Definition 6 *Let* Γ *be any set of* TL*-formulas,* M_1 *and* M_2 *any models of* Γ*, and* Δ *any set of defaults.* M_2 *is* **fulfillment-preferred** *to* M_1 *wrt* Γ *and* Δ *(written* $M_1 \sqsubset_F M_2$*) iff*

1. M_1 *and* M_2 *are both discriminant models over the same domain* D*,*
2. $E_1^2 \neq E_2^1$*,*
3. $\forall \delta : \exists \langle \delta, d \rangle \in E_1^2 \cup E_2^1 \rightarrow F_{M_1}(\delta) \subseteq F_{M_2}(\delta)$.

The example in Figure 1 shows that fulfillment-preference is needed to deal with conflicting default-chains. This is due to the fact that wherever you "cut" a chain of defaults you get away with one exception. If you take into account fulfillers, however, the models that fulfill as many defaults in the chain as possible are preferred.

Now we can finally define Δ-preference between models:

Definition 7 *Let* Γ *be any set of* TL*-formulas,* M_1 *and* M_2 *any models of* Γ*, and* Δ *any set of defaults.* M_2 *is* **Δ-preferred to** M_1 *wrt* Γ *(written* $M_1 \sqsubset_\Delta M_2$*) iff*

1. $M_1 \sqsubset_E M_2$ *or*
2. $M_2 \not\sqsubset_E M_1 \wedge M_1 \sqsubset_F M_2$.

A model M_1 *is* **Δ-preferred** *iff there is no model* M_2 *such that* $M_1 \sqsubset_\Delta M_2$.

Δ-preference has the desirable property of being bounded (cf. [Shoham 88]):

Proposition 2 *Let* Γ *be any set of* TL*-formulas and* Δ *any set of defaults.* $\sqsubset_\Delta$ *is anti-symmetric and bounded, i.e. there is no infinite sequence of models* $M_1 \sqsubset_\Delta M_2 \sqsubset_\Delta \ldots$.

Proof: Anti-Symmetricity follows from the condition $E_1^2 \neq E_2^1$ in definitions 5 and 6.
Boundedness follows from the fact that if $M_1 \sqsubset_\Delta M_2$ then there is at least one exception $\langle \delta, d \rangle$ in E_1^2, i.e., there is one exception in M_1 which is no exception in M_2. Exceptions are thus mininimzed along the precedence $\sqsubset_\Delta$ until they finally reach a local minimum. Thus no infinite chain is possible. □

From this we can immediately proof the following proposition:

Proposition 3 *Let* Γ *be any satisfiable set of* TL*-formulas,* Δ *any set of defaults, and* α_1, α_2 *any descriptions. The set of* Δ*-preferred models for* Γ *is non-empty.*

Proof: By assumption Γ is satisfiable, i.e., the set of models of Γ is not empty. Since these models can not form a cyclic or infinite chain of Δ-preferences (Proposition 2), at least one of them must be a preferred model. □

3 Δ-ENTAILMENT

Having specified the preference relation between models we will now define Δ-entailment. We will separate between Δ-entailment of descriptions and of subsumptions, however. The reason for doing so will become obvious below.

3.1 DESCRIPTIONS

We define Δ-entailment in the usual manner, except that we restrict it to descriptions:

Definition 8 *Let* Γ *be any set of* TL*-formulas,* α *any description and* Δ *any set of defaults.* Γ **Δ-entails** α *(written* $\Gamma \models_\Delta \alpha$*) iff all* Δ*-preferred models of* Γ *are models of* α.

[4] Something similar to the condition of closure by default inference in Reiter's Default Logic (default extensions are closed by default inferences).

When motivating the integration of defaults into TL we said that their main function was to increase the information that can be derived about an object. Since the set of preferred models is a subset of the set of models it is obvious that $\Gamma \models \alpha$ implies $\Gamma \models_\Delta \alpha$. More interesting than this lower bound of the Δ-entailed descriptions is the upper bound. In particular we want to know whether a satisfiable set of of TL-formulas can become inconsistent when defaults are added.

Proposition 4 *Let Γ be any satisfiable set of TL-formulas, Δ any set of defaults, and α_1, α_2 assertional formulas. Thus, if $\Gamma \cup \{\alpha_1, \alpha_2\}$ is inconsistent then $\Gamma \not\models_\Delta \alpha_1$ or $\Gamma \not\models_\Delta \alpha_2$.*

Proof: Assume the opposite, i.e. both α_1 and α_2 are Δ-entailed. That is all preferred models of Γ satisfy α_1 and α_2. Since $\Gamma \cup \{\alpha_1, \alpha_2\}$ is inconsistent there cannot be any such model, thus the set of preferred models of Γ must be empty. But from Proposition 3 we know that Γ, being satisfiable, has at least one preferred model. □

After these general results let us consider some more examples to get a grasp of the details of Δ-entailment. We mentioned above that defaults "inherit" from rules the problems of disjunction and lifting. Let us begin with the disjunction problem.

Since we defined rules $c_1 \rightarrow c_2$ as equivalent to subsumptions $c_1 \sqsubseteq c_2$, rules behave like material implications. In other words $c_1 \rightarrow c_2$ is equivalent to $\top \rightarrow \neg c_1 c_2 \sqcup$. Rules thus introduce implicit disjunction and negation, even if the underlying TL does not contain disjunction or negation at all. As a consequence, a simple forward-chaining algorithm testing which premises of rules are subsumers of a concept and then adding there conclusions is incomplete (it misses contraposition, for example). To avoid this problem the semantics of rules can be weakened as shown in [Schild 89]. Thus, the TL-systems BACK [Quantz, Kindermann 90] and CLASSIC [Brachman et al. 91] treat rules not as material implications but as forward-chaining rules. LOOM [MacGregor 91], on the other hand, treats rules (though not defaults) as material implications, and we think that this is the intuitively correct interpretation. As a consequence, we currently investigate the extension of rule-inferences in BACK.

Now due to our definition of exceptions, defaults also resemble material implications: an object is an exception to a default δ if it is an instance of δ_p but not of δ_c—in other words, it is no exception if it is either an instance of $\neg\delta_p$ or of δ_c. A default δ is thus weakly equivalent to the default $\top \rightsquigarrow \neg\delta_p \sqcup \delta_c$—only weakly because of our definition of specificity (for details see Section 3.3). As a consequence, contraposition is valid in certain circumstances as illustrated by the following example:

		o	E_1
γ_1: o :: c_2	M_1:	c_1	$\{o\}$
δ_1: $c_1 \rightsquigarrow \neg c_2$	$*M_2$:	$\bar{c}_1$	$\emptyset$
$M_1 \sqsubset_E M_2$, $\models_\Delta$ o :: $\neg c_1$			

Another illustration for the disjunctive conception of defaults is provided by the following example:

			o	o	E_1	E_2
γ_1:	o :: c_1	$*M_1$:	c_2	c_3	$\emptyset$	$\emptyset$
δ_1:	$c_1 \sqcap c_2 \rightsquigarrow c_3$	M_2:	c_2	$\bar{c}_3$	$\{o\}$	$\emptyset$
δ_2:	$c_1 \sqcap \neg c_2 \rightsquigarrow c_3$	$*M_3$:	$\bar{c}_2$	c_3	$\emptyset$	$\emptyset$
$\models_\Delta$	o :: c_3	M_4:	$\bar{c}_2$	$\bar{c}_3$	$\emptyset$	$\{o\}$

3.2 LIFTING

The second heritage from rules is the lifting problem. Subsumption between concepts can be "lifted" to subsumption between quantified concepts (cf. [Royer, Quantz 92]):

$$c_1 \sqsubseteq c_2 \models \forall r : c_1 \sqsubseteq \forall r : c_2$$
$$c_1 \sqsubseteq c_2 \models \geq nr : c_1 \sqsubseteq \geq nr : c_2$$
$$c_1 \sqsubseteq c_2 \models \leq nr : c_2 \sqsubseteq \leq nr : c_1$$

The "exchange" of c_1 and c_2 in the last entailment is due to $\leq n$ being a $MON\downarrow$ quantifier, whereas $\forall$ and $\leq n$ are $MON\uparrow$ (cf. [Quantz 92]).

This is important since algorithms have to "expand" the concepts when computing consequences of rules. Thus, even though the concept $\forall$ r:c_1 is not subsumed by c_1 itself, it contains c_1 embedded in a quantificational restriction. To deal with this issue, rule application can be either restricted to explicitly mentioned role-rillers like in CLASSIC. It seems that the results presented in [Baader, Hollunder 92] and [Donini et al. 92] can be used to specify a semantics for this conception of rules (Peter Patel-Schneider, personal communication). Alternatively, rules can be applied to embedded concepts as specified in [Schild 89] an implemented in BACK.

The relevance of this issue for defaults is illustrated by the following Δ-entailments for $\Delta = \{c_1 \rightsquigarrow c_2\}$:

$$o :: \forall r : c_1 \models_\Delta o :: \forall r : c_2$$
$$o :: \geq nr : c_1 \models_\Delta o :: \geq nr : c_2$$
$$o :: \leq nr : c_2 \models_\Delta o :: \leq nr : c_1$$

Thus, in the following example o is known to have a filler of type c_1 at r. Though there is no object-name denoting this filler we get the expected Δ-entailment, since we use the domain D and not the object-names themselves to determine the exceptions. We thus avoid skolemization on the sematic level and the severe problems investigated in [Baader, Hollunder 92].

			d	E_1
γ_1:	o :: ≥ 1 r:c_1	$*M_1$:	c_2	$\emptyset$
δ_1:	$c_1 \rightsquigarrow c_2$	M_2:	$\bar{c}_2$	$\{d\}$
$\models_\Delta$	o :: ≥ 1 r:c_2			

To deal with the problem on a syntactic level we propose the following lifting strategy:

Definition 9 *Let δ_1, δ_2 be any defaults. δ_2 is* **directly lifted** *from δ_1 (written $\delta_1 \xrightarrow{l} \delta_2$) iff*

1. $\delta_2 = qr : \delta_{1_p} \rightsquigarrow qr : \delta_{1_c}$ *where* q *is* $\geq n$ *or* $\forall$, *i.e.* $MON\uparrow$, *or*

2. $\delta_2 = qr : \delta_{1_c} \rightsquigarrow qr : \delta_{1_p}$ *where* q *is* $\leq n$, *i.e.* $MON\downarrow$.

δ_2 *is* **lifted** *from* δ_1 *(written* $\delta_1 \xrightarrow{l^*} \delta_2$*) iff*

1. $\delta_1 = \delta_2$ *or*

2. $\exists \delta_3 : \delta_1 \xrightarrow{l^*} \delta_3 \wedge \delta_3 \xrightarrow{l} \delta_2$.

Now the obvious question to ask is whether the "lifting completion" of a set of defaults changes its meaning. To answer this question we need to specify, however, when two sets of defaults have the same meanings.

3.3 DEFAULT SUBSUMPTION

We have so far restricted our attention to the Δ-entailment of descriptions. We did so because we think that these are the inferences most needed in actual applications. It is, however, also interesting to consider the general properties of default theories, in particular to characterize the precise differences between strict entailment and default entailment. We will thus indicate how our theory can be investigated in this direction and we will sketch the impact of this investigation on default subsumption.

Let us begin with the notion of *redundancy*. Intuitively, a set of defaults Δ' is redundant wrt a set of defaults Δ iff $\Delta \cup \Delta'$ yields the same consequences as Δ. There are, however, three possibilities to define this formally:

1. Consider a given set Γ and define Δ_2 as being redundant wrt. Δ_1 iff Γ Δ-entails the same descriptions from Δ_1 and $\Delta_1 \cup \Delta_2$.
2. Define redundancy by quantifying over Γ, i.e., all Γ have to Δ-entail the same descriptions from Δ_1 and $\Delta_1 \cup \Delta_2$.
3. Additionally quantify over super sets of defaults, i.e., all Γ have to Δ-entail the same descriptions from all Δ_3 and $\Delta_3 \cup \Delta_2$, where $\Delta_1 \subseteq \Delta_3$.

Obviously, each definition imposes stronger conditions on redundancy than its predecessor. The last definition corresponds to Morreau's definition of default entailment and has some desirable logical properties (e.g., monotonicity) [Morreau 92]. We suppose, however, that the strong restrictions imposed by it will hardly yield any redundancies at all. We make no choice for a particular definition of redundancy in this paper. Rather we indicate the relevant aspects for redundancy.

The most obvious aspect is the *conceptual content* of a default. Thus a default δ' is conceptually redundant wrt. a default δ, iff all Γ yield the same strict consequences when unified with $\{\delta_p \sqsubseteq \delta_c\}$ and with $\{\delta_p \sqsubseteq \delta_c, \delta'_p \sqsubseteq \delta'_c\}$. Note that $\top \rightsquigarrow \neg\delta_p \sqcup \delta_c$, $\neg\delta_c \rightsquigarrow \neg\delta_p$, and $\forall\, r{:}\delta_p \rightsquigarrow \forall\, r{:}\delta_c$ are all conceptually redundant wrt. δ.

The conceptual content of a default is not the only aspect relevant for redundancy, however. Consider for example the default $\delta_1 = c_1 \rightsquigarrow c_2 \sqcap c_3$. The defaults $\delta_2 = c_1 \rightsquigarrow c_2$ and $\delta_1 = c_1 \rightsquigarrow c_3$ are conceptually redundant wrt. δ_1 but they clearly change the Δ-entailed descriptions. The reason is that they can be refuted separately while this is obviously not possible for δ_1:

$$o :: c_1 \sqcap \neg c_2 \quad \not\models_{\{\delta_1\}} \quad o :: c_3$$

$$o :: c_1 \sqcap \neg c_2 \quad \models_{\{\delta_1, \delta_2, \delta_3\}} \quad o :: c_3$$

Morreau uses this example to distinguish between theories that view defaults as inference tickets and normalcy theories [Morreau 92]. Obviously, our theory belongs to the former group.

Another important issue for redundancy is the precedence relation between defaults. Thus, adding a default will have consequences, if it precedes some other default. Consequently, there is an important difference between $\delta_1 = \top \rightsquigarrow \neg c_1 \sqcup c_2$ and $\delta_2 = c_1 \rightsquigarrow c_2$. Since we defined precedence via subsumption of premises, δ_2 can precede other defaults, whereas δ_1 cannot.

To sum up, redundancy of defaults depends on several factors. In particular, lifted defaults are not necessarily redundant. This fact is important for the syntactic characterizations provided in Sections 4 and 5, since it is one source of incompleteness.

Let us now turn to the question of default subsumption. The notion of redundancy can obviously used to define default subsumption. But first let us consider why we restricted Definition 8 of Δ-entailment to descriptions. A model satifies $c_1 \sqsubseteq c_2$ iff $[\![c_1]\!]^{\mathcal{I}} \subseteq [\![c_2]\!]^{\mathcal{I}}$. Now one certainly would expect entailment of the default subsumption $c_1 \sqsubseteq c_2$ from a default $c_1 \rightsquigarrow c_2$. But as soon as we have a set of TL-formulas entailing the existence of at least one exception to the default, $[\![c_1]\!]^{\mathcal{I}} \subseteq [\![c_2]\!]^{\mathcal{I}}$ would be false in all models and thus also in the preferred ones. We thus need to define default subsumption separately.

One possible definition of default subsumption reduces it to redundancy: or more precisely to one of the possible definitions of redundancy. We could say that Γ Δ-entails $c_1 \sqsubseteq c_2$ for a set of defaults Δ iff $c_1 \rightsquigarrow c_2$ is redundant wrt. Δ. We would thus get different results of default subsumption depending on the version of redundancy we choose.

Another way of tackling default subsumption is to ignore the descriptions in Γ. This is in line with the general reluctance against the influence of descriptions on term subsumption. Usually, the information available on an object o is *not* used in the classification of a concept containing r:o. We can thus say that a subsumption θ is Δ-entailed by Γ iff all preferred models of the subsumptions in Γ are models of θ.

Obviously, this gives the required entailment of $c_1 \sqsubseteq c_2$ from a default $c_1 \rightsquigarrow c_2$. The definition yields unintuitive results in case of conflicting defaults, however. Since a set of subsumptions has no existential impact at all, a model can always be constructed as having the empty set as its domain. Since this model has no exceptions at all it is

always a preferred model, and in particular it is preferred to any model with at least one exception. Thus conflicting defaults yield incoherent concepts, even when a precedence holds between them. We would thus get $c_2 \sqsubseteq \bot$ as default subsumption of $c_2 \sqsubseteq c_1$ for the defaults $\{c_1 \rightsquigarrow \neg c_3, c_2 \rightsquigarrow c_3\}$.

The most obvious definition for default subsumption, and the one most in line with the approach presented here, consists in reducing default subsumption completely to Δ-entailment of descriptions. We thus say that $c_1 \sqsubseteq c_2$ is Δ-entailed from Γ iff $o :: c_2$ is Δ-entailed from $\Gamma \cup \{o :: c_1\}$.

4 CONFLICT RESOLUTION

We will now turn our attention towards the computational task. So far we have illustrated the effects of our semantics by presenting small examples. In order to construct algorihms, however, we need to find general statements expressing the effects in case of conflicting defaults. In general we need a characterization of Δ-entailment in syntactic terms, thereby abstracting from the underlying model-theoretic semantics. We see this section as a first step in this direction. We emphasize, however, that our syntactic characterizations are sound but not complete. We will sketch how completeness might be achieved, but we can also imagine to take the incomplete syntactic characterization as a basis for implementation.

Let us start with two general remarks concerning conflicts in TL. First, conflicts do not necessarily derive from two defaults but can be caused by an arbitrary number of defaults. To see why, just consider the concept $\forall\, r{:}c \sqcap \forall\, r{:}\neg c \sqcap \geq 1\ r{:}\top$. Here three components together cause the inconsistency of the concept: two value-restrictions for r which are disjoint and a minimum-restriction which asserts the existence of at least one filler for r. Note that these components are not mutually disjoint, however. In general we can construct inconsistencies caused by an arbitrary number of components by arbitrarily embedding the conflicting components as value restrictions:
$\forall\, r_1{:}\forall\, r_2{:}c \sqcap \forall\, r_1{:}\forall\, r_2{:}\neg c \sqcap \forall\, r_1{:} \geq 1\ r_2{:}\top \sqcap \geq 1\ r_1{:}\top$.

The other important aspect of conflicts in TL is that they are not necessarily local: due to the impact of value restrictions at role-fillers, two object descriptions can be locally consistent at two objects, but cause a conflict if one object is a role-filler at another object. In the following example we thus have a conflict between two defaults at different object:

		o_2	E_1	E_2
γ_1: $o_1 :: c_1 \sqcap r{:}o_2$	M_1:	c_3	$\emptyset$	$\{o_2\}$
γ_2: $o_2 :: c_2$	M_2:	$\bar{c}_3$	$\{o_1\}$	$\emptyset$
δ_1: $c_1 \rightsquigarrow \forall\, r{:}c_3$				
δ_2: $c_2 \rightsquigarrow \neg\, c_3$				

We begin with a characterization of the strict information available at an object for a given set of TL-formulas:

Definition 10 *Let Γ be any set of* TL*-formulas and o any object-name. We define the* **most specific description** *of o wrt Γ (written $\mu_\Gamma(o)$) as the concept c for which $\Gamma \models o :: c$ and there is no c' such that $\Gamma \models o :: c' \wedge \Gamma \models c' \sqsubseteq c$.*

Now the most obvious case of conflict resolution occurs when a default is not applied because it contradicts strict information. Furthermore, we are not interested in defaults yielding no additional information wrt. strict information. We thus define applicability of a default at an object:

Definition 11 *Let Γ be any set of* TL*-formulas, δ any default, and o any object-name. δ is* **applicable** *at o wrt Γ iff $\Gamma \not\models \mu_\Gamma(o) \sqsubseteq \delta_p \sqcap \neg\delta_c$ and $\Gamma \not\models \mu_\Gamma(o) \sqsubseteq \neg\delta_p \sqcup \delta_c$.*

As pointed out above, defaults must be considered from a "global" perspective. We therefore introduce the notion of a *default space* in which sets of defaults are assigned to several objects at once:

Definition 12 *Let Γ be any set of* TL*-formulas and Δ any set of defaults. A* **default space** *Σ over Δ is a set $\{\langle o_1, \Delta_1\rangle \ldots \langle o_n, \Delta_n\rangle\}$. $\langle o_i, \Delta_i\rangle$ $(1 \leq i \leq n)$ is called a* **component** *of Σ and $\langle o_i, \delta\rangle$ for $\delta \in \Delta_i$ an* **atom** *of Σ. There is at most one component $\langle o_i, \Delta_i\rangle$ for each o_i in Σ, and for all atoms $\langle o, \delta\rangle$ δ is applicable at o and $\exists \delta' \in \Delta : \delta' \xrightarrow{I^o} \delta$.*

A default space Σ is a **subspace** *of a default space Σ' iff $\Sigma \neq \Sigma'$ and for each component $\langle o, \Delta\rangle$ in Σ there is a component $\langle o', \Delta'\rangle$ in Σ' such that $\Delta \subseteq \Delta'$.*

Having thus defined default spaces we now have to define the result of applying a default space to a set of TL-formulas (note that the conceptual value of a default δ is $\neg\delta_p \sqcup \delta_c$):

Definition 13 *Let Γ be any set of* TL*-formulas, Δ any set of defaults, and Σ any default space over Δ. The* **application** *of Σ to Γ is defined as*

$$\Sigma(\Gamma) \stackrel{\text{def}}{=} \Gamma \cup \{o :: \sqcap\Delta : \langle o, \Delta\rangle \in \Sigma\},$$

where $\sqcap\{\delta_1, \ldots, \delta_n\} \stackrel{\text{def}}{=} (\neg\delta_{1p} \sqcup \delta_{1c}) \sqcap \ldots \sqcap (\neg\delta_{np} \sqcup \delta_{nc})$.

So far we did not say anything about the consistency of default spaces. We will therefore turn our attention towards conflicts:

Definition 14 *Let Γ be any set of* TL*-formulas, Δ any set of defaults, and Σ any default space over Δ. Σ is a* **conflict** *wrt Γ iff $\Sigma(\Gamma)$ is inconsistent. Otherwise Σ is* **consistent** *wrt Γ. Σ is a* **minimal conflict** *iff it is a conflict and all subspaces of Σ are consistent.*

We can then proof that Δ-entailment complies with the intuition that defaults not involved in any conflict should be applied:

Proposition 5 *Let Γ be any set of* TL*-formulas, Δ any set of defaults, δ any default in Δ, and o any object-name. If δ is applicable at o and there is no minimal conflict over Δ in which $\langle o, \delta\rangle$ is an atom, then $\Gamma \models_\Delta o :: \neg\delta_p \sqcup \delta_c$.*

Proof: Assume otherwise, i.e., there is a preferred model M_1 in which $[o]^{\mathcal{I}_1} \notin [\neg\delta_p \sqcup \delta_c]^{\mathcal{I}_1}$. Clearly, $[o]^{\mathcal{I}_1}$ is an exception to δ in M_1. Sine δ is applicable at o there are models M_2, however, which are like M_1, except that in them $[o]^{\mathcal{I}_2} \in [\neg\delta_p \sqcup \delta_c]^{\mathcal{I}_2}$ and thus $[o]^{\mathcal{I}_2}$ is no exception to δ. Since M_1 is a preferred model, i.e. $M_1 \not\preceq_\Delta M_2$, there must thus be other exceptions in the M_2 making up for the exception in M_1 (see Proposition 1).

Assume that there is exactly one such exception $[o']^{\mathcal{I}}$ to a default δ' in all the M_2. Since we characterized the M_2 as differing from M_1 only wrt. $[o]^{\mathcal{I}_2} \in [\neg\delta_p \sqcup \delta_c]^{\mathcal{I}_2}$ the exception must depend on this property, i.e., whenever $[o]^{\mathcal{I}} \in [\neg\delta_p \sqcup \delta_c]^{\mathcal{I}}$ then $[o']^{\mathcal{I}} \notin [\neg\delta'_p \sqcup \delta'_c]^{\mathcal{I}}$. But this means that $\Gamma \cup \{o :: \neg\delta_p \sqcup \delta_c, o' :: \neg\delta'_p \sqcup \delta'_c\}$ is inconsistent and hence that $\{\langle o, \{\delta\}\rangle, \langle o', \{\delta'\}\rangle\}$ is a minimal conflict.

To end the proof, consider the case in which there are different exceptions in the M_2. Since there must be one in each model their disjunction is induced by $[o]^{\mathcal{I}} \in [\neg\delta_p \sqcup \delta_c]^{\mathcal{I}}$, hence all together are inconsistent and we get a conflict. (If the M_2 contain more than one additional exception each we can construct several conflicts.)

Finally, assume that the exceptions do not correspond to object-names but are implicit role-fillers. But then we get the conflict by taking a lifted default at o. □

We now investigate the case of conflicting defaults. Note that a minimal conflict can be resolved by removing one single default:

Definition 15 *Let Σ be any minimal conflict. Any atom $\langle o, \delta\rangle$ of Σ is a* **resolution** *of Σ.*

The essence of conflict resolution is then to decide, which default has to be removed. Here we can use again our notion of default precedence defined in terms of specificity:

Definition 16 *Let Γ be any set of* TL*-formulas, Σ any minimal conflict, and $\rho_1 = \langle o_1, \delta_1\rangle$, $\rho_2 = \langle o_2, \delta_2\rangle$ two resolutions of Σ. We say that ρ_1 is* **preferred** *to ρ_2 wrt Σ (written $\rho_1 \prec_\Sigma \rho_2$) iff $\delta_2 \prec_\Gamma \delta_1$. A resolution ρ_1 is* **preferred** *wrt Σ iff there is no ρ_2 such that $\rho_2 \prec_\Sigma \rho_1$.*

Before continuing let us consider some examples of conflicts. We begin with three defaults applicable at one object, such that one pair of defaults is compatible, while the other two pairs form conflicts.

In the first example δ_2 conflicts with a preceded default δ_1 which in turn conflicts with δ_3:

	o	o	E_1	E_2	E_3
γ_1: $c_2 \sqsubseteq c_1$	$M_1 : c_4$	c_5	$\emptyset$	$\{o\}$	$\{o\}$
γ_2: $o :: c_2 \sqcap c_3$	$M_2 : c_4$	$\bar{c}_5$	$\{o\}$	$\{o\}$	$\emptyset$
δ_1: $c_1 \rightsquigarrow \neg c_4 \sqcap c_5$	$M_3 : \bar{c}_4$	c_5	$\{o\}$	$\emptyset$	$\{o\}$
δ_2: $c_2 \rightsquigarrow \neg c_4$	$*M_4 : \bar{c}_4$	$\bar{c}_5$	$\{o\}$	$\emptyset$	$\emptyset$
δ_3: $c_3 \rightsquigarrow \neg c_5$					
$M_2 \sqsubset_E M_4$, $M_3 \sqsubset_E M_4$					
$M_1 \sqsubset_E M_4$ (because of specificity)					
$\models_\Delta o :: \neg c_4 \sqcap \neg c_5$					

We can interpret this example by saying that δ_1 is cancelled by δ_2 and that δ_1 and δ_3 remain active. In the following example δ_3 does not conflict with δ_1 but instead with the preceding default δ_2:

	o	o	E_1	E_2	E_3
γ_1: $c_2 \sqsubseteq c_1$	$*M_1 : c_4$	c_5	$\{o\}$	$\emptyset$	$\{o\}$
γ_2: $o :: c_2 \sqcap c_3$	$M_2 : c_4$	$\bar{c}_5$	$\{o\}$	$\{o\}$	$\emptyset$
δ_1: $c_1 \rightsquigarrow \neg c_4$	$M_3 : \bar{c}_4$	c_5	$\emptyset$	$\{o\}$	$\{o\}$
δ_2: $c_2 \rightsquigarrow c_4 \sqcap c_5$	$*M_4 : \bar{c}_4$	$\bar{c}_5$	$\emptyset$	$\{o\}$	$\emptyset$
δ_3: $c_3 \rightsquigarrow \neg c_5$					
$M_2 \sqsubset_E M_4$, $M_3 \sqsubset_E M_4$					

In this example none of the defaults is active: δ_1 is cancelled by δ_2 whereas δ_2 and δ_3 neutralize each other. Finally, we consider an example which is just like the previous one except that δ_3 here precedes δ_2.

	o	o	E_1	E_2	E_3
γ_1: $c_2 \sqsubseteq c_1$	$M_1 : c_4$	c_5	$\{o\}$	$\emptyset$	$\{o\}$
γ_2: $c_3 \sqsubseteq c_2$	$M_2 : c_4$	$\bar{c}_5$	$\{o\}$	$\{o\}$	$\emptyset$
γ_3: $o :: c_3$	$M_3 : \bar{c}_4$	c_5	$\emptyset$	$\{o\}$	$\{o\}$
δ_1: $c_1 \rightsquigarrow \neg c_4$	$*M_4 : \bar{c}_4$	$\bar{c}_5$	$\emptyset$	$\{o\}$	$\emptyset$
δ_2: $c_2 \rightsquigarrow c_4 \sqcap c_5$					
δ_3: $c_3 \rightsquigarrow \neg c_5$					
$M_2 \sqsubset_E M_4$, $M_3 \sqsubset_E M_4$					
$M_1 \sqsubset_E M_4$ (because of specificity)					
$\models_\Delta \neg c_5 :: \neg c_4 \sqcap o$					

In this example δ_2 is cancelled by δ_3 and can thus not cancel δ_1 anymore—δ_1 and δ_3 are active.

We will now formalize our intuitive talk about active, cancelled, and neutralized defaults. Our formal definitions will not cover the above examples and hence the semantics completely. In general we will cancel more defaults than actually necessary thus guaranteeing soundness but lacking completeness:

Definition 17 *Let Γ be any set of* TL*-formulas, o any object-name, and δ any default applicable at o wrt Γ. We say that δ is* **active** *at o iff there is no conflict Σ for which $\langle o, \delta\rangle$ is a preferred resolution.*

Note that this definition misses the fact that a default can be active even if it forms a preferred resolution, namely when the default preceding it is cancelled in some other conflict. In other words, not every default which should be active according to the semantics is active according to this definition.

The above definition gives a sound syntactic characterization, however, as shown by the following proposition:

Proposition 6 *Let Γ be any set of* TL*-formulas, Δ any set of defaults, δ any default in Δ, and o any object-name. If δ is active at o then $\Gamma \models_\Delta o :: \neg\delta_p \sqcup \delta_c$.*

Proof: The beginning of the proof is identical to the proof for Proposition 5 and we end up with a minimal conflict Σ in which $\langle o, \delta\rangle$ is an atom.

In each of the M_2 there is thus one atom $\langle o', \delta' \rangle$ of Σ such that $[o']^{\mathcal{I}_2}$ is an exception to δ' in M_2(but $[o']^{\mathcal{I}_1}$ is no exception to δ' in M_1). Since no M_2 can be preferred to M_1, no δ' can precede δ. Hence $\langle o, \delta \rangle$ must be a preferred resolution of Σ. But this is a contradiction to the assumption that δ is active at o. □

The opposite of active defaults are cancelled defaults:

Definition 18 *Let Γ be any set of* TL*-formulas, o any object-name, and δ any default applicable at o wrt Γ. We say that δ is* **cancelled** *at o iff $\langle o, \delta \rangle$ is a preferred resolution for a conflict Σ and there is another resolution ρ_2 of Σ such that $\rho_2 \prec_\Sigma \rho_1$.*

Note that this definition misses the fact that a cancelled default can become active again when the cancelling default is cancelled itself. In other words, some defaults are cancelled according to this definition without being cancelled according to the semantics.

A third category comprises the neutralized defaults. The distinction to cancelled defaults can be illustrated by the following example:

			o	o	E_1	E_2
γ_1:	$o :: c_1 \sqcap c_2$	$*M_1$:	c_3	c_4	$\emptyset$	$\{o\}$
δ_1:	$c_1 \rightsquigarrow c_3 \sqcap c_4$	M_2:	c_3	$\bar{c}_4$	$\{o\}$	$\{o\}$
δ_2:	$c_2 \rightsquigarrow \neg c_3 \sqcap c_4$	$*M_3$:	$\bar{c}_3$	c_4	$\{o\}$	$\emptyset$
$\models_\Delta$	$o :: c_4$	M_4:	$\bar{c}_3$	$\bar{c}_4$	$\{o\}$	$\{o\}$

If two defaults neutralize each other they do not loose all their force. Rather their disjunction is still "active". This is a consequence, of defining Δ-entailment via satisfaction in all preferred models. A cancelled default is not fulfilled in any preferred model; an active default is fulfilled in all preferred models; in each preferred model at least one of a set of neutralized defaults is fulfilled.

Definition 19 *Let Γ be any set of* TL*-formulas, Δ any set of defaults, and Σ any consistent default space over Δ. Σ is a* **neutralization** *iff it is a subspace of a minimal conflict Σ' wrt. Γ containing all atoms $\langle o, \delta \rangle$ of Σ' for which δ is neither active nor cancelled at o.*

Obviously, since we have too many cancelled defaults we get too few neutralizations. But again we have a sound characterization of Δ-entailment:

Proposition 7 *Let Γ be any set of* TL*-formulas, Δany set of defaults, Σ any neutralization, and o any object-name. If $\langle o, \{\delta_1, \ldots, \delta_n\} \rangle$ is a component of Σ and $n > 1$ then $\Gamma \models_\Delta o :: \neg\delta_{1_p} \sqcup \delta_{1_c} \sqcup \ldots \sqcup \neg\delta_{n_p} \sqcup \delta_{n_c}$.*

Proof: Since Σ is a subspace of a minimal conflict there are two possibilities.

First assume that $\{\langle o, \{\delta_1, \ldots, \delta_n\} \rangle\}$ is itself the minimal conflict. Since $n > 1$ there are at least two defaults δ_1, δ_2 forming the conflict at o. Now assume a model M_1 in which $[o]^{\mathcal{I}_1} \notin [\neg\delta_{1_p} \sqcup \delta_{1_c} \sqcup \neg\delta_{2_p} \sqcup \delta_{2_c}]^{\mathcal{I}_1}$, and in which $[o]^{\mathcal{I}_1}$ is thus an exception to both δ_1 and δ_2. Clearly, M_1 cannot be preferred since we can construct a model in which $[o]^{\mathcal{I}}$ is only an exception to one of these.

To end the proof assume that Σ contains more than just $\langle o, \{\delta_1, \ldots, \delta_n\} \rangle$. But then $\{\langle o, \{\delta_1, \ldots, \delta_n\} \rangle\}$ is not a conflict and hence a model in which $[o]^{\mathcal{I}}$ is an exception to all $\delta_1, \ldots, \delta_n$ cannot be preferred either. □

Given a set of TL-formulas and a set of defaults we can thus collect the defaults active at objects and the neutralized defaults. These collections can then be turned into descriptions:

Definition 20 *Let Γ be any set of* TL*-formulas and Δ any set of defaults. We define*

$$A_\Delta \stackrel{\text{def}}{=} \{o :: \neg\delta_p \sqcup \delta_c : \delta \text{ is active at } o\} \text{ and}$$

$$N_\Delta \stackrel{\text{def}}{=} \{o :: \neg\delta_{1_p} \sqcup \delta_{1_c} \sqcup \ldots \sqcup \neg\delta_{n_p} \sqcup \delta_{n_c} : \langle o, \{\delta_1, \ldots, \delta_n\} \rangle \text{is a component of a neutralization} \wedge n > 1\}$$

We can now summarize our sound syntactic characterization of Δ-entailment by using Propositions 6 and 7:

Proposition 8 *Let Γ be any set of* TL*-formulas, Δ any set of defaults, and α any description.*
If $\Gamma \cup A_\Delta \cup N_\Delta \models \alpha$ then $\Gamma \models_\Delta \alpha$.

Our syntactic characterization is incomplete since we have too many cancelled and too few active defaults. We could fix this shortcoming by refining the corresponding definitions. There is another source of incompleteness, however, namely lifting. In order to remedy this shortcoming we would have to characterize a lifting-completion for a set of defaults. Alternatively, we could integrate some kind of skolemization. Due to the arguments presented in Section 3.2 and [Baader, Hollunder 92], however, we think this will be a rather non-trivial task.

We think that our syntactic characterization has some justification on its own: it is comparatively easy and complies with the semantics in case of binary, transitive conflicts. We think that these kinds of conflicts are the ones most likely to occur in realistic applications though this intuition has to be checked empirically, of course.

5 ANTERIORITY

So far we considered specificity as the only precedence criterion between defaults. Obviously, there are other precedence criteria that might be taken into account for conflict resolution. Thus it might be argued that cancelling a default involved in many conflicts is preferrable to cancelling one involved in only one conflict. To take another example, if an object is strictly known to be an instance of a premise of a default this default should have more impact than one where the object is only defeasibly known to be an instance of the premise. We will thus consider exemplarily the integration of an additional precedence criterion into our semantics: the criterion of *anteriority* aims at capturing

the application ordering between defaults which depending on the subsumption hierarchy induced by strict knowledge. It provides an implicit forward chaining interpretation of defaults — a view closer to the interpretation of defaults in Reiter's Default Logic.

Note that from a methodological point of view, we still carry on with a multi-criteria approach. As many preference criteria as wanted can be integrated, adopting for example the aggregation principle of Grosof [Grosof 91]. In his approach, given a partially ordered family of preference relations, the global preference between two entities is given by the strongest (according to the partial ordering) individual preference relation in which the two entities are strictly comparable.

5.1 ANTERIORITY PREFERENCE

The following example illustrates the effect of anteriority:

$$\begin{aligned}
\Gamma &= \{c_3 \sqsubseteq c_4, c_1 \sqsubseteq c_4, c_3 \sqcap c_5 \sqsubseteq \bot, o :: c_1 \sqcap c_2\} \\
\delta_1 &= c_2 \rightsquigarrow c_3 \\
\delta_2 &= c_4 \rightsquigarrow c_5 \\
M_1 &= \{c_1(d), c_2(d), c_4(d), c_5(d)\} \\
M_2 &= \{c_1(d), c_2(d), c_3(d), c_4(d)\} \\
E_2^1 &= \{\langle \delta_1, d\rangle\}, E_1^2 = \{\langle \delta_2, d\rangle\} \\
F_2^1 &= \{\langle \delta_2, d\rangle\}, F_1^2 = \{\langle \delta_1, d\rangle\}
\end{aligned}$$

Both M_1 and M_2 are preferred models. However, M_2 is intuitively preferable to M_1 because it contains *globally* more specific knowledge than M_1. The conclusion of δ_1 is more specific than the premisse of δ_2, δ_2 should be preferentially cut in the conflict.

In a first approximation, one default δ will be said anterior to one set of default links Δ whenever the premise of δ is monotonically entailed by both the premises and the consequents of the defaults of Δ. However, this condition miss the object reasoning level. Indeed default links can be "chained" even if they don't tell about the same object. For example, in the example below, we want δ_1 to be anterior to δ_2:

$$\begin{aligned}
\Gamma &= \{o' :: c_1, o :: r : o'\} \\
\delta_1 &= c_1 \rightsquigarrow c_2 \\
\delta_2 &= \geq 1 r c_2 \rightsquigarrow c_3
\end{aligned}$$

There is a logical implication between o' :: c_2 (consequence of δ_1) and o :: $\geq$ 1 r c_2. In case of conflicts, δ_1 would be preferred, again because it provides with most specific descriptions.

Consequently, anteriority has also to abstract description dependencies at the object level, namely dependencies between a quantified description and a non quantified one at a role filler[5]. We thus propose the following definition of anteriority-preference.

[5]Note that due to the restricted negation in BACK V5 we do not consider negative dependencies.

Definition 21 *Let Γ be a set of terminological formulas. Let us define the following description dependency rules for every role term r in Γ :*

$$\begin{aligned}
c &\rightarrow c \ (identity) && (15)\\
c &\rightarrow \geq nr : c \ (generalization) && (16)\\
\leq nr : c &\rightarrow c \ (no - particularization) && (17)\\
\forall r : c &\rightarrow c \ (all - particularization) && (18)
\end{aligned}$$

Definition 22 *Let Γ be any set of* TL*-formulas. Let Δ be a set of defaults. The anteriority relation wrt Γ and Δ, written $\prec_A$, is the least partial pre-ordering on Δ verifying the following conditions:*

1. *$\delta_1 \prec_A \delta_2$ iff δ_1 belongs to a* least *subset Δ' of Δ such that:*

$$\Gamma \models \sqcap_{\delta' \in \Delta'}(\delta'_p \sqcap \ell(\delta'_c)) \sqsubseteq \delta_{2p} \quad (19)$$

$$\Gamma \not\models \sqcap_{\delta' \in \Delta'}(\delta'_p) \sqsubseteq \delta_{2p} \quad (20)$$

   where for every default δ' of Δ', $\ell(\delta'_c)$ follows from δ'_c by application of the description dependency rules.

2. *If δ' is a lifted form of δ_1 and $\delta' \prec_A \delta_2$ then $\delta_1 \prec_A \delta_2$.*

Definition 23 *Let Γ be any set of* TL*-formulas, M_1 and M_2 any discriminant models of Γ with the same domain D, and Δ any set of Defaults. M_2 is* **anteriority-preferred** *to M_1 wrt Γ and Δ (written $M_1 \sqsubset_A M_2$) iff for every $\langle \delta_1, o_1\rangle$ in E_1^2 , there exists $\langle \delta_2, o_2\rangle$ in E_2^1 such that $\delta_1 \prec_A \delta_2$.*

In the following we call "anteriority preference semantics" the preference defined by exception, fulfillment and anteriority, in this priority order.

5.2 A FIXED-POINT CHARACTERIZATION

Anteriority can be used to define a stratification of the set of defaults. The stratification is given by the least *total pre-ordering* which extends the partial pre-ordering $\prec_A$. The first stratum is constituted by the defaults of Δ which are minimal for $\prec_A$ (in short, $\prec_A$-minimal), the second stratum by the set of defaults minimal for $\prec_A$ in Δ minus the first stratum, etc...

Definition 24 *Let Γ be a set of* TL*-formulas. Let Δ be a set of defaults. We define a partition of Δ into* **anteriority strata** *(Δ_i , i in {1,n}) as follows.*

1. *Δ_0 is the empty set.*

2. *For every $i \leq n$, Δ_{i+1} is the subset of $\prec_A$-minimal defaults of $\Delta \setminus \cup_{j=1}^{j=i}(\Delta_j)$.*

We thus get immediately a *computation strategy* for deriving the formulas entailed by Γ and Δ wrt anteriority preference semantics. The idea is to consider the defaults of the first stratum, to fire a maximal subset of them consistently, by solving conflicts in the stratum according to specificity, then to iterate on the second stratum and so on.

The question is whether this intuitive procedure is sound and complete wrt anteriority semantics. We only are able to give some partial answers.

First we have no reason to expect completeness. Because the stratification ordering extends the initial anteriority partial pre-ordering, a stratification-based procedure may introduce new preferences between conflicting defaults. [6] Secondly, we need to cope with inferences at implicit objects. One approach for that is some kind of "skolemization", i.e., introducing explicit names for the objects implicitly defined by existential descriptions.

Skolemization is problematic because, as known from first-order logic, it does not preserve logical equivalence (not even monotonic).(It only preserves (un)satisfiability, which makes it adequate for refutation methods such as Resolution Principles, but not for direct proof methods.) Adequat skolemization would thus introduce skolems in a way which guarantees at least monotonic logical equivalence. This involves searching for "minimal characteristic descriptions" on which skolemization could be safely done.

Instead we will rather follow the approach of lifting defaults adequately, as discussed in Section 3.3. The problem of default lifting consists in the fact that due to precedences between defaults, lifted defaults can be non-redundant. To cope with this problem a default δ should be lifted at an object o only if the most specific description of o is non redundant with other descriptions at objects o_i at which the default is directly applicable. This motivates the following definitions.

Definition 25 *Let Γ be a set of* TL*-formulas, δ a default, o an object name, $\delta_\exists = \geq n\ r{:}\delta_p \rightsquigarrow \geq n\ r\delta_c$, and $\delta_\forall = \forall\ r{:}\delta_p \rightsquigarrow \forall\ r{:}\delta_c$.*
δ is **firable** *at o wrt Γ iff $\Gamma \models \mu_\Gamma(o) \sqsubseteq \delta_p$.*
δ can be **existentially lifted** *at o wrt Γ iff :*

1. *$\delta_\exists$ is firable at o wrt Γ;*
2. *there are no other objects $(o_i, i=1...n)$ such that, for every i, $\Gamma \models o_i :: r{:}o$ and δ is firable at o_i.*

δ can be **universally lifted** *at o wrt Γ iff :*

1. *$\delta_\forall$ is firable at o wrt Γ ;*
2. *there is no set of objects $(o_i, i \text{ in } I)$, and no descriptions α_i at o_i such that $\Gamma \models o_i :: r{:}o$, $\Gamma \models \alpha_i$ and $o :: \mu_\Gamma(o)$ follows from the descriptions $\{ \alpha_i, o_i :: r{:}o, i \text{ in } I \}$ alone.*

We now try to give a syntactic characterization of preferential entailment, which looks similar to the notion of extension of Reiter's Default Logic. The essential difference comes from the particular handling of lifted default links.

[6] It is easy to show that the definition of anteriority-preference is monotonic wrt to the anteriority pre-ordering between defaults. Thus extending the anteriority relation diminishes the number of preferred models. See [Cayrol et al. 92] for details on this point.

Definition 26 *A set E of* TL*-formulas is a default extension of Γ and Δ iff E is a maximal consistent set of* TL*-formulas containing Γ verifying the following closure conditions :*

1. *If δ in Δ is firable at an object o wrt to E, then $o :: \delta_c$ belongs to E ;*
2. *If δ in Δ can be lifted at an object o wrt to E, then $o :: \delta'_c$ belongs to E, where δ' is the lifted form of δ.*

The above characterization is obviously a non-monotonic fixed-point definition, due to non-monotonicity of the lifting conditions in 25. We get, however, some partial monotonicity wrt anteriority. If a default can be fired or lifted at o wrt to E and Δ_i , then it can be also wrt to Δ_j for every j>i. The key point is that, if addition of a new default δ_2 gives a new description which defeats a firability condition at δ_1 then necessarily $\delta_2 \prec_A \delta_1$ (anteriority was intended to reflect all possible logical dependencies between defaults). This is enough to justify a stepwise computation of default extensions according to the anteriority strata. Resulting default extensions then can be shown to be anteriority preferred, in a sense similar to the definition of anteriority preference between models (knowledge sets are considered instead of models). The proof relies upon general results stated in [Cayrol et al. 92].

The problem of computing default extensions on a given anteriority stratum still remains, however. In particular a notion of exception-preferred extensions has to be formalized (in a way similar as above) to prove the following conjecture: a formula is preferentially entailed wrt to anteriority semantics iff it is monotonically entailed from every exception- and anteriority-preferred default extension.

6 CONCLUSION

We have presented a preference semantics for defaults in terminological logics. This semantics contains specificity as a precedence criterion between defaults and thus allows overwriting of defaults. The semantics yields disjunctive skepticism in case of non-resolvable conflicts.

In order to find a connection between the semantics and the computation task we have investigated two different approaches. First we introduced the notion of default spaces and conflict. We have then defined the notions of active, cancelled, and neutralized defaults which provide a sound syntactic characterization of default entailment. The incompleteness is due to the fact that cancelled defaults can become active when the cancelling defaults are cancelled themselves. Furthermore, lifted defaults which are necessary to deal with implicit objects are redundant only in certain contexts. To obtain a complete syntactic characterization the definitions of active and cancelled defaults as well as the conditions for default lifting have to be refined. This is a non-trivial task, however.

Our first approach thus uses a top-down strategy where the set of all possible conflicts is determined in advance and a

resolution procedure is applied to construct preferred consistent default spaces. Our second approach contains a fixed point characterization of default inferences. This exploits a third preference criterion, anteriority, capturing some forward chaining interpretation of defaults links. The characterization is monotonic wrt. to strict anteriority, which suggests an incremental computation wrt to anteriority strata. Here again, soundness of the procedure is conjectured (with partial preferences, completeness is false in general).

We see our syntactic characterizations as a good starting point for prototypical implementations in TL-systems. These implementations should be used to gather experiences concerning the use of defaults in applications. We see in particular a need to evaluate the adequacy of the precedence criterion between defaults, the frequency and complexity of conflicts, and the effect of forward chaining of default rules on the overall performance. It might turn out that complex conflicts which can be constructed in theory do not occur frequently in practical applications and that hence incomplete algorithms are useful enough.

Moreover an investigation on the usability of defaults in machine translations [Schmitz, Quantz 92], has shown that in this application defaults should be seen as constraints rather than inference rules. Instead of applying defaults and resolving conflicts the system should return the defaults violated by certain descriptions. The exceptions induced by several possible descriptions of an object are then evaluated outside the TL-system to find the most acceptable solution. A similar strategy of external conflict resolution is assumed in the LOOM applications (Bob MacGregor, personal communication).

Acknowledgements

This work was supported by the Commission of the European Communities and is part of the ESPRIT Project 5210 AIMS. We would like to thank Klaus Schild for many valuable ideas and criticisms. We also want to thank Bob MacGregor, Michael Morreau, Peter Patel-Schneider, and Oli-Kai Paulus for discussions on issues treated in this paper. One of the authors also profited from a discussion with Lin Padgham, Berhard Nebel, and Bob MacGregor during the 1991 Workshop on Terminological Logics in Dagstuhl.

References

[Baader, Hollunder 92] F. Baader, B. Hollunder, "Embedding Defaults into Terminological Knowledge Representation Formalisms", to appear in [Nebel et al. 92]

[Bossu, Siegel 85] G. Bossu, P. Siegel, "Saturation, Nonmonotonic Reasoning and the Closed-World Assumption", *Artificial Intelligence* 25, 13–63, 1985

[Brachman et al. 91] R. Brachman, D.L. McGuiness, P.F. Patel-Schneider, L. Alperin Resnick, A. Borgida, "Living with CLASSIC: When and How to Use a KL-ONE-like Language", in [Sowa 91], 401–456

[Cayrol et al. 92] C. Cayrol, V. Royer, C. Saurel, "Management of Preferences in Assumption-Based Reasoning", *Information Processing and Management of Uncertainty*, Palma de Mallorca, Spain, July 1992.

[Donini et al. 92] F. Donini, M. Lenzerini, D. Nardi, A. Schaerf, W. Nutt, "Adding Epistemic Operators to Concept Languages", to appear in [Nebel et al. 92]

[Gelfond et al. 89] M. Gelfond, H. Przymusinska, T. Przymusinski, "On the Relationship between Circumscription and Negation as Failure" *Artificial Intelligence* 38, 75–94, 1989

[Grosof 91] B.N. Groshof, "Generalizing Priorization", *Proceedings of KR'91*, 289–300, 1991

[Kindermann 90] C. Kindermann, "Class Instances in a Terminological Framework–an Experience Report", in H. Marburger (ed.), *Proc. of GWAI-90*, Berlin: Springer, 48–57, 1990

[Kindermann 92] C. Kindermann, "Retraction in Terminological Knowledge Bases", to appear in *ECAI'92*

[MacGregor 91] R. MacGregor, "Using a Description Classifier to Enhance Deductive Inference", in *Proceedings Seventh IEEE Conference on AI Applications*, Miami, Florida, February, 1991, 141–147

[McCarthy 86] J. McCarthy, "Applications of Circumscription to Formalizing Common-Sense Knowledge", *Artificial Intelligence* 28, 89–116 1986

[Morreau 92] M. Morreau, *Conditionals in Philosophy and Artificial Intelligence*, Amsterdam Dissertation, to appear

[Nebel et al. 92] B. Nebel, C. Rich, W. Swartout, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR92)*, San Mateo: Morgan Kaufmann, 1992

[Padgham 89] L. Padgham, *Non-Monotonic Inheritance for an Object Oriented Knowledge Base*, Linköping Studies in Science and Technology, Dissertation No. 213, 1989

[Pfahringer 89] B. Pfahringer, "Integrating Definitions and Defaults", *Workshop on Term Subsumption Languages*, 1989

[Quantz 92] J. Quantz, "How to Fit Generalized Quantifiers into Terminological Logics", to appear in *ECAI-92*

[Quantz, Kindermann 90] J. Quantz, C. Kindermann, *Implementation of the BACK-System Version 4*, KIT-Report 78, Technische Universität Berlin, 1990

[Royer, Quantz 92] V. Royer, J.J. Quantz, "Deriving Inference Rules for Terminological Logics", to appear in *JELIA'92*

[Schild 89] K. Schild, *Towards a Theory of Frames and Rules*, KIT-Report 76, Technische Universität Berlin, 1989

[Schmitz, Quantz 92] B. Schmitz, J.J. Quantz, "Defaults in Machine Translation", to appear in the Proceedings of the Colloquium on Defaults in Knowledge and Language Processing

[Shoham 88] Y. Shoham, *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*, Cambridge: MIT Press, 1988

[Sowa 91] J. Sowa (Ed.), *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, San Mateo: Morgan Kaufman, 1991

Embedding Defaults into Terminological Knowledge Representation Formalisms

Franz Baader and Bernhard Hollunder
German Research Center for AI (DFKI)
Stuhlsatzenhausweg 3, 6600 Saarbrücken 11, Germany
e-mail: ⟨*last name*⟩@dfki.uni-sb.de

Abstract

We consider the problem of integrating Reiter's default logic into terminological representation systems. It turns out that such an integration is less straightforward than we expected, considering the fact that the terminological language is a decidable sublanguage of first-order logic. Semantically, one has the unpleasant effect that the consequences of a terminological default theory may be rather unintuitive, and may even vary with the syntactic structure of equivalent concept expressions. This is due to the unsatisfactory treatment of open defaults via Skolemization in Reiter's semantics. On the algorithmic side, we show that this treatment may lead to an undecidable default consequence relation, even though our base language is decidable, and we have only finitely many (open) defaults. Because of these problems, we then consider a restricted semantics for open defaults in our terminological default theories: default rules are only applied to individuals that are explicitly present in the knowledge base. In this semantics it is possible to compute all extensions of a finite terminological default theory, which means that this type of default reasoning is decidable.

1 Introduction

Terminological representation systems are used to represent the taxonomic and conceptual knowledge of a problem domain in a structured and well-formed way. To describe this kind of knowledge, one starts with atomic concepts (unary predicates) and roles (binary predicates), and defines more complex concepts using the operations provided by the concept language of the particular formalism. In addition to this concept description formalism, most of these systems also have an assertional component. One can for example state that an individual is an instance of a concept, or that two individuals are connected by a role.

In terminological representation formalisms, the concept descriptions are interpreted as universal statements, which means, unlike frame languages, they do not allow for exceptions. As a consequence, the system can use descriptions to automatically insert concepts at the proper place in the taxonomy (classification), and it can use the facts stated about individuals to deduce to which concepts they must belong (realization). For example, one could define the concept *Mammal* as an *Animal* that feeds its young with *Milk*, where *feeds-young-with* is used as a role. If the concept *Platypus*[1] is defined as an *Animal* that *lives-in* the *Water*, feeds its young with *Milk*, and *reproduces* with *Eggs*, then the system will recognize that *Platypus* is a subconcept of *Mammal*.

However, commonsense reasoning is often based on assumptions that may ultimately be shown to be false. In our example, one might want to assume by default that *Mammals reproduce Viviparously*. Only if it is known that a specific mammal reproduces with eggs, should this assumption be cancelled. If one wants to use terminological systems for this kind of commonsense reasoning, one needs a formalism that can handle such default assumptions, but does not destroy the definitional character of concept descriptions—because otherwise the advantage of automatic concept classification, etc., would be lost (see [3]). Besides the general arguments for the importance of reasoning with defaults, which can be found in the nonmonotonic reasoning literature, the need for embedding defaults into terminological representation formalisms is also substantiated by the fact that this is an important item on the wish list of users of terminological representation systems (see e.g. [20]).

Several existing terminological systems, such as BACK [18], CLASSIC [4], K-REP [14], LOOM [17], or SB-ONE

[1] We are taking this as our exceptional animal, in view of the fact that last IJCAI was in Australia, and not in the Antarctic.

[12], have been or will be extended to provide the user with some kind of default reasoning facilities. However, as the designers of these systems themselves point out, these approaches usually have an ad hoc character, and are not equipped with a formal semantics. For example, defaults in the FAME system, which is built using K-Rep, "will not be complete (or even consistent)" ([14], p.11) unless the user is very careful when using them. In CLASSIC, "a limited form of defaults can be represented with the aid of rules and test functions." However, the user is warned to "use this trick with extreme caution" ([4], p.45,46).

Our arguments for the importance of default extensions for terminological representation languages so far were given from the viewpoint of the terminological systems community. However, these investigations may also be of interest for research in nonmonotonic reasoning itself. Most nonmonotonic reasoning formalisms (e.g. Reiter's default logic [23], Circumscription [15]) use full first-order predicate logic as their base language. In this general form, the formalisms are usually highly undecidable (see e.g. [23] Theorem 4.9). For this reason, work on decision procedures for decidable subcases was mostly restricted to propositional logic (see e.g. [11]), thus leaving the wide gap between propositional logic and full first-order logic almost unexplored. Since most terminological representation languages can be viewed as decidable subclasses of first-order logic—but are nevertheless much more expressive than propositional logic—they can serve as interesting test cases for nonmonotonic reasoning formalisms. We shall see that this not only applies for algorithmic, but also for semantic considerations.

We shall here consider the problem of integrating Reiter's default logic into a terminological representation formalism. This treatment of defaults in terminological systems has already been proposed by Brachman and Schmolze [5], but to the best of our knowledge, this proposal was never followed up. Reiter's default rule approach seems to fit well into the philosophy of terminological systems because most of them already provide their users with a form of "monotonic" rules. These rules can be considered as special default rules where the justifications—which make the behaviour of default rules nonmonotonic—are absent.

At first sight, one might think that, from a semantic point of view, the proposed integration should be unproblematic. In fact, the terminological representation language we shall consider (see Section 2) is a sublanguage of first-order logic, and Reiter's semantics has been formulated for full first-order logic. However, on closer inspection it turns out that one runs into severe problems, due to the unsatisfactory treatment of open defaults by Skolemization (see Section 3).

A similar problem arises when considering the integration from the algorithmic point of view. In the abstract of their paper on how to compute extensions for default logic, Junker and Konolige [10] write that their method is applicable if the default theory "consists of a finite number of defaults and premises and classical derivability for the base language is decidable." A related formulation can be found in the abstract of Schwind and Risch's paper on the same topic [25]. Since our base language is decidable, and we certainly do not want to have infinitely many default rules, these methods seem to apply in our case. However, a closer look at the papers reveals that by "a finite number of defaults" it is meant "a finite number of *closed* defaults." But the default rules we want to consider are *open* defaults. In fact, as already pointed out by Reiter ([23], p.115) "the genuinely interesting cases involve open defaults." In Section 4 we shall show that, with our (decidable) terminological language as base language, a finite set of premises and open defaults may lead to an undecidable default consequence problem, if the open defaults are treated as proposed by Reiter ([23], Section 7.1).

Because of the semantic as well as algorithmic problems posed by Reiter's treatment of open defaults, we shall consider a restricted semantics for open defaults in our integration: default rules are only applied to individuals that are *explicitly* present in the assertional part (ABox) of the knowledge base. Though one may thus lose some intuitive default inferences, this treatment of default rules is akin to the treatment of the monotonic rules in terminological systems such as CLASSIC.

With this restricted semantics, a finite set of open defaults stands for a set of closed defaults that is finite as well. Thus the above-mentioned methods of Schwind and Risch and of Junker and Konolige can be applied to compute extensions (see Section 5). In order to make these methods more efficient, one has to solve certain algorithmic problems for the terminological language. For Junker and Konolige's methods one has to find minimal proofs for assertional facts—which can be seen as an abduction problem for ABoxes—and for Schwind and Risch's method one must find maximal consistent sets of assertional facts. In Section 6 we shall point out how the tableaux-based methods for assertional reasoning developed in our group ([8, 1]) can be modified to solve these problems.

2 The Representation Formalisms

First we shall briefly review the terminological language $\mathcal{ALCF}$ [9] and Reiter's default logic. Then terminological default logic is defined as the specialization of default logic to $\mathcal{ALCF}$. Finally an example will illustrate why Reiter uses Skolemization in his semantics for open default theories.

2.1 The terminological language $\mathcal{ALCF}$

Terminological knowledge representation formalisms can be used to define the relevant concepts of a problem domain (terminological knowledge), and to describe objects of this domain with respect to their relation to concepts and their interrelation with each other (assertional knowledge). Depending on which constructs are allowed for building concept descriptions we get different terminological languages. In the present paper we restrict our attention to the language $\mathcal{ALCF}$.

Definition 2.1 *The* terminological part *of the language $\mathcal{ALCF}$ consists of the following concept description formalism. The* concept terms *of this formalism are built from* concept, role and attribute names *using the constructors* conjunction *($C \sqcap D$)*, disjunction *($C \sqcup D$)*, negation *($\neg C$)*, exists-restriction *($\exists R.C$)*, value-restriction *($\forall R.C$), and* agreement *($u \doteq v$). Here C, D stand for concept terms, R for a role or attribute name, and u, v for finite sequences of attribute names.*
The assertional part *of our language allows us to assert facts concerning particular objects. These objects are referred to by* individual names, *and we can state that an object belongs to a concept (written $C(a)$), or that two objects are related by a role or attribute (written $R(a,b)$). Here a, b stand for individual names, C for a concept term, and R for a role or attribute name. A finite set of such facts is called an* ABox.

The semantics of an ABox can either be given directly by defining interpretations and models, or by a translation into first-order logic. In order to make the fact explicit that we are dealing with a sublanguage of first-order logic, we choose the second option.

Concept names are considered as symbols for unary predicates, and role and attribute names as symbols for binary predicates. Consequently, concept names A are translated into (atomic) formulae $A(x)$ with one free variable, and role and attribute names R into (atomic) formulae $R(x,y)$ with two free variables. The attributes have to be interpreted as partial functions, which can be expressed by a formula $\forall x, y, z\colon (f(x,y) \wedge f(x,z) \rightarrow y = z)$ for each attribute name f.

Concept terms are also translated into formulae with one free variable. The semantics of conjunction, disjunction, and negation are defined in the obvious way, i.e., $(C \sqcap D)(x) := C(x) \wedge D(x), (C \sqcup D)(x) := C(x) \vee D(x), (\neg C)(x) := \neg C(x)$. For value-restrictions we define $(\forall R.C)(x) := \forall y\colon (R(x,y) \rightarrow C(y))$, and the semantics of exists-restrictions is given by $(\exists R.C)(x) := \exists y\colon (R(x,y) \wedge C(y))$. Let $u = f_1 \cdots f_m$, and $v = g_1 \cdots g_n$ be sequences of attributes. The agreement construct built from these sequences is translated into the formula $(u \doteq v)(x) := \exists y_1, \ldots, y_m, z_1, \ldots z_n\colon (f_1(x,y_1) \wedge \ldots f_m(y_{m-1}, y_m) \wedge g_1(x,z_1) \wedge \ldots g_n(z_{n-1}, z_n) \wedge y_m = z_n)$.

The individual names of the Abox are considered as constant symbols. In terminological systems one usually has a *unique name assumption*, which can be expressed by the formulae $a \neq b$ for all distinct individual names a, b. The formula corresponding to the assertional fact $C(a)$ (resp. $R(a,b)$) is obtained by replacing the free variable(s) in the formula corresponding to C (resp. R) by a (resp. a, b). To sum up, an ABox is translated into a set of first order formulae consisting of the translations of the ABox facts, the formulae expressing unique name assumption, and the formulae expressing that attributes are partial functions.

The basic inference service for ABoxes is called *instantiation*. It answers the question of whether (the translation of) a given ABox fact $C(a)$ is a (logical) consequence of (the translation of) a given ABox $\mathcal{A}$. If the answer is yes we say that *a is an instance of C with respect to $\mathcal{A}$* ($\mathcal{A} \models C(a)$). Algorithms which solve this inference problem have, for example, been described in [8, 1].

2.2 Reiter's default logic

Reiter [23] deals with the problem of how to formalize nonmonotonic reasoning by introducing nonstandard, nonmonotonic inference rules, which he calls default rules. A *default rule* is any expression of the form

$$\frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma},$$

where α, β_1, ..., β_n, γ are first-order formulae. Here α is called the *prerequisite* of the rule, $\beta_1, \ldots, \beta_n$ are its *justifications*, and γ is its *consequent*. For a set of default rules $\mathcal{D}$, we denote the sets of formulae occurring as prerequisites, justifications, and consequents in $\mathcal{D}$ by $Pre(\mathcal{D})$, $Jus(\mathcal{D})$, and $Con(\mathcal{D})$, respectively.

A default rule is *closed* iff $\alpha, \beta_1, \ldots, \beta_n, \gamma$ do not contain free variables. A *default theory* is a pair $(\mathcal{W}, \mathcal{D})$ where $\mathcal{W}$ is a set of closed first-order formulae (the world description) and $\mathcal{D}$ is a set of default rules. A default theory is *closed* iff all its default rules are closed.

Intuitively, a *closed* default rule can be applied, i.e., its consequent is added to the current set of beliefs, if its prerequisite is already believed and all its justifications are consistent with the set of beliefs. Formally, the consequences of a *closed* default theory are defined with reference to the notion of an *extension*, which is a set of deductively closed first-order formulae defined by a fixed point construction (see [23], p.89). In general, a default theory may have more than one extension, or even no extension. Depending on whether one wants to employ skeptical or credulous reasoning, a closed formula δ is a *consequence of a closed default theory* iff it is in all extensions or if it is in at least one extension of the theory. In general, this consequence

relation is not even recursively enumerable (see [23], Theorem 4.9).

Reiter also gives an alternative characterization of an extension, which we shall use, in a slightly modified way, as the definition of extension. Here and in the following, $Th(\Gamma)$ stands for the deductive closure of a set of formulae Γ.

Definition 2.2 *Let E be a set of closed formulae, and $(\mathcal{W},\mathcal{D})$ be a closed default theory. We define*

$$E_0 := \mathcal{W}$$

and for all $i \geq 0$

$$E_{i+1} := E_i \cup \{\gamma \mid \alpha : \beta_1, \ldots, \beta_n/\gamma \in \mathcal{D}, \alpha \in Th(E_i), \text{ and } \neg\beta_1, \ldots, \neg\beta_n \notin Th(E)\}.$$

Then $Th(E)$ is an extension of $(\mathcal{W},\mathcal{D})$ iff

$$Th(E) = \bigcup_{i=0}^{\infty} Th(E_i).$$

Note that the extension $Th(E)$ to be constructed by this iteration process occurs in the definition of each iteration step. Since we are only adding consequents of defaults during the iteration, any extension $Th(E)$ of $(\mathcal{W},\mathcal{D})$ is of the form $Th(\mathcal{W} \cup Con(\widehat{\mathcal{D}}))$ for a subset $\widehat{\mathcal{D}}$ of $\mathcal{D}$. This set is called the set of *generating defaults* of the extension. Another easy consequence of the definition is that $(\mathcal{W},\mathcal{D})$ has an inconsistent extension iff $\mathcal{W}$ is inconsistent.

Reiter defines *extensions of arbitrary default theories* $(\mathcal{W},\mathcal{D})$, i.e., default theories with open defaults, as follows. First, the formulae of $\mathcal{W}$ and the consequents of the defaults are Skolemized (see [23], Section 7). Second, a set $\mathcal{D}'$ of closed default rules is generated by taking all ground instances (over the initial signature together with the newly introduced Skolem functions) of the defaults of $\mathcal{D}$. Now E is an *extension* of $(\mathcal{W},\mathcal{D})$ iff E is an extension of the closed default theory $(\mathcal{W}',\mathcal{D}')$, where $\mathcal{W}'$ is the Skolemized form of $\mathcal{W}$. The reason for Skolemizing before building ground instances will be explained by an example in Subsection 2.4.

2.3 Terminological default theories

A *terminological default theory* is a pair $(\mathcal{A},\mathcal{D})$ where $\mathcal{A}$ is an ABox and $\mathcal{D}$ is a finite set of default rules whose prerequisites, justifications, and consequents are concept terms. Obviously, since ABoxes can be seen as sets of closed formulae, and since concept terms can be seen as formulae with one free variable,[2] terminological default theories are subsumed by Reiter's notion of an open default theory.

[2]The concept terms occurring in one rule are assumed to have identical free variables.

However, as for ABox reasoning without defaults, we are not interested in arbitrary formulae as consequences of a terminological default theory $(\mathcal{A},\mathcal{D})$, but only in assertional facts of the form $C(a)$, where a is an individual name occurring in the original ABox $\mathcal{A}$.

2.4 Why is Skolemization necessary ?

The following example shows that intuitively valid consequences would get lost if one did not Skolemize. Suppose that our ABox consists of the fact that *Tom* has some child who is a doctor, i.e., $\mathcal{A} = \{(\exists child.doctor)(Tom)\}$. By default we want to conclude that doctors usually are rich persons, and usually have children who are doctors. Thus $\mathcal{D}$ consists of the default rules

$$\frac{doctor : rich\text{-}person}{rich\text{-}person} \quad \text{and} \quad \frac{doctor : \exists child.doctor}{\exists child.doctor}.$$

Skolemization of the world description $\mathcal{A}$ yields $\mathcal{A}' = \{child(Tom, Bill), doctor(Bill)\}$, where *Bill* is a new Skolem constant, whereas Skolemization of the consequent of the second default yields a unary Skolem function, say *child-of*. It is easy to see that the corresponding closed default theory has exactly one extension, and that this extension contains the assertional facts that *Tom* has a rich child and a grandchild who is a doctor, i.e., $(\exists child.rich\text{-}person)(Tom)$, and $(\exists child.\exists child.doctor)(Tom)$. Intuitively, this comes from the fact that the closed defaults obtained by instantiating our open defaults with the Skolem constant *Bill* are applicable. Without these ground instances, the above facts could not have been deduced by default. To deduce by default that the grandchild of *Tom* is not only a doctor, but also a rich one, the first default has to be instantiated by the term *child-of*(*Bill*).

3 Problems Caused by Skolemization

In addition to the problem that Skolemization usually destroys the nice compositional character of our concept formulae, it is also problematic for more severe reasons to be presented below. We shall give three examples which demonstrate that Reiter's treatment of open defaults is problematic, from an intuitive as well as a formal point of view.

Our *first example* shows that the Skolemization of the world description may lead to counterintuitive consequences of the default theory. Consider the following concept term which can be used to express that an adult man is married to a woman or is a bachelor

$$(\exists spouse.Woman) \sqcup Bachelor.$$

We assume that our ABox asserts that the individual *Tom* belongs to this concept term, and that he is married to the woman *Mary*. In addition, we take the following default (without prerequisite)

$$\frac{: \neg Woman}{\neg Woman},$$

which corresponds to a still-prevailing male chauvinism in linguistic usage. In order to know with what individuals this default has to be instantiated, we have to Skolemize our ABox facts. Translated into traditional first-order syntax, these facts yield the world description

$$\{(\exists y\colon spouse(Tom, y) \wedge Woman(y)) \vee Bachelor(Tom),\ spouse(Tom, Mary),\ Woman(Mary)\}.$$

The Skolemized version of the first formula is

$$(*)\quad (spouse(Tom, Gordy) \wedge Woman(Gordy)) \vee Bachelor(Tom),$$

where *Gordy* is introduced as a new Skolem constant. Because of the disjunction in this formula, our Skolemized world description does not imply $Woman(Gordy)$. Thus the chauvinistic default can fire, and we get $\neg Woman(Gordy)$. Together with the formula $(*)$ this yields $Bachelor(Tom)$ as a consequence of our default theory, which is rather surprising since our ABox actually contains a female spouse of *Tom*.

As already pointed out by Poole, the reason for this strange behaviour comes from that fact that "we have lost the context of what the Skolem constants represent" ([21], p.907), in our case the context that *Gordy* was originally introduced to stand for a female spouse of *Tom*. Poole proposes to keep track of this context by using Hilbert's ϵ-symbol.

Although Poole's approach may avoid the problem in the above example, it is of no avail in our next examples. These examples demonstrates that, due to the problems caused by Skolemization, the consequences of a default theory depend on the ***syntactic*** form of the world description, i.e., for identical sets of open defaults, logically equivalent world descriptions may lead to different results.

In our ***second example*** we consider concept terms $C_1 := \exists R.(A \sqcap B)$ and $C_2 := \exists R.A$ where R is a role name and A, B are concept names. Obviously, if we assert that an individual a is in the first term this implies that it is in the second one as well. For this reason, the ABoxes $\mathcal{A}_1 := \{C_1(a)\}$ and $\mathcal{A}_2 := \{C_1(a), C_2(a)\}$ are logically equivalent. When Skolemizing the first ABox, we get a single new Skolem constant b which is R-related to a and lies in $A \sqcap B$, whereas when Skolemizing the second ABox we get two Skolem constants c and d, both R-related to a, but where c lies in $A \sqcap B$ and d lies in A. Now consider the (open) default $A : \neg B/\neg B$. For the Skolemized version of $\mathcal{A}_1$, this default is instantiated with a, b, whereas for the Skolemized version of $\mathcal{A}_2$ it is instantiated with a, c, d. Obviously, the default rule *cannot* fire for b and c, because their being in $A \sqcap B$ is inconsistent with its justification. On the other hand, this default rule *can* be applied to d, because being in A is consistent with being in $\neg B$. For this reason, d is put into $\neg B$, which shows that the Skolemized version of $\mathcal{A}_2$ has $(\exists R.\neg B)(a)$ as a default consequence, whereas this fact cannot be deduced by default from the Skolemized version of $\mathcal{A}_1$. Technically, the reason for this behaviour is due to the fact that, before the application of the default, the individuals c and d might be identical (which is the reason why the two ABoxes are logically equivalent) whereas this is no longer possible after the default has been applied.

The ***third example*** is similar to the second. It is quite obvious that the concept terms $\exists R.(A \sqcup B)$ and $(\exists R.A) \sqcup (\exists R.B)$ are equivalent. Let $\mathcal{A}_1$ be an ABox where a is asserted to be in the first concept term, and $\mathcal{A}_2$ one where a is asserted to be in the second concept term. When using a standard Skolemization method, the first ABox yields one new Skolem constant, and the second ABox yields two. Now it is easy to see that the corresponding instantiations of the default rule $A \sqcup B : C/C$ can only fire for the Skolemized version of the first ABox. Consequently, we have a in $\exists R.C$ as a default consequence of the first ABox, but not of the second one, even though these two ABoxes are equivalent.

Lifschitz [13] proposes a treatment of open defaults which avoids Skolemization by working with classes of models instead of sets of formulae in the definition of default extensions. Obviously, working with models means that logically equivalent formulae must yield the same results. This shows that Lifschitz's approach can overcome the problem pointed out in the previous two examples, even though it was not motivated by the problems connected with Skolemization (see footnote 1 in [13]: "Skolemization ... is irrelevant for this discussion.") Lifschitz's motivation was to make it possible to derive by default universally quantified formulae of the form $\forall x\colon C(x)$, which is not possible with Reiter's approach, but which is not necessary in our context (because the terminological inference service is only meant to derive new ABox facts, i.e., formulae of the form $C(a)$). From our point of view, the main problem of Lifschitz's approach is that working with models means that it becomes even harder to get algorithms for computing extensions. Another problem of his approach is that one gets rather unexpected consequences, due to the fact that models of different cardinality are treated separately. For example, assume that one has formulae ≥ 3 and ≤ 2 expressing that a model has at least 3 and at most 2 elements, respectively, which would, for example, be available in concept languages allowing for number-restrictions and a universal role, i.e., a role U that satisfies $\forall x, y\colon U(x, y)$. The default theory consisting of an empty world description and the closed defaults

$$\frac{\leq 2 :}{C(a)} \quad \text{and} \quad \frac{\geq 3 :}{C(a)}$$

has $C(a)$ as consequence, which means that this approach makes a case analysis with respect to the cardinality of models. But for other cases, Lifschitz's ap-

proach still does not make case analysis. For example, the theory consisting of an empty world description and the closed defaults

$$\frac{A(a):}{C(a)} \quad \text{and} \quad \frac{\neg A(a):}{C(a)}$$

does not have $C(a)$ as a consequence.

4 An Undecidability Result

In addition to the semantic problems caused by Skolemization, we shall now show that, for our base language $\mathcal{ALCF}$, this treatment of open defaults also leads to an undecidable default consequence relation, even though $\mathcal{ALCF}$ is decidable. This is achieved by reducing the word problem for semigroups [22] to the consequence problem of a default theory.

Let Σ be a finite alphabet, and let $R = \{(u_1, v_1), \ldots, (u_n, v_n)\}$ be a finite set of relations presenting a semigroup over Σ. In the following we shall treat the elements of Σ as attribute names. The semigroup presentation is used to define a finite set of open defaults as follows. For any $f \in \Sigma$ and for any relation $(u_i, v_i) \in R$ we have defaults

$$\frac{A:}{\forall f.A} \quad \text{and} \quad \frac{A:}{u_i \doteq v_i}.$$

If we want to decide whether the words u, v are equivalent with respect to R, we take the ABox $\mathcal{A}_{u,v} := \{A(a), (u \doteq u)(a), (v \doteq v)(a)\}$ as our world description.

Proposition 4.1 *With respect to the set of defaults induced by Σ and R, the ABox fact $(u \doteq v)(a)$ is a default consequence of $\mathcal{A}_{u,v}$ iff u and v are equivalent with respect to R.*

Intuitively, the world description puts a into A, and asserts sequences of attributes u, v starting from a. The implicit individuals lying on these sequences are made explicit by Skolemization. The first type of defaults puts all individuals reachable from a by a sequence of attributes into A, and the second type identifies individuals which can be reached by the respective sequences u_i and v_i from an individual in A, thus simulating application of relations from R. (It should be noted that the consequences of this second type of defaults are also responsible for the introduction of new implicit individuals.)

Since a formal proof of the proposition is straightforward but rather tedious, we shall just illustrate it by an example. Consider the semigroup presentation $R = \{(fg, gf)\}$ over the alphabet $\Sigma = \{f, g\}$. This presentation is transformed into the default rules

$$\frac{A:}{\forall f.A}, \quad \frac{A:}{\forall g.A}, \quad \text{and} \quad \frac{A:}{fg \doteq gf}.$$

Obviously, the words fgg and ggf are equivalent with respect to R. If we want to obtain this equivalence as a consequence of applying the above default rules, we take the Abox $\mathcal{A}_{fgg,ggf} = \{A(a), (fgg \doteq fgg)(a), (ggf \doteq ggf)(a)\}$ as our world description.

Translated into first-order logic and then Skolemized, this ABox yields the world description

$$\begin{aligned}\{\ & A(a),\\ & f(a,b_1) \wedge g(b_1,b_2) \wedge g(b_2,b_3),\\ & g(a,c_1) \wedge g(c_1,c_2) \wedge f(c_2,c_3),\\ & \forall x,y,z\colon (f(x,y) \wedge f(x,z) \rightarrow y = z),\\ & \forall x,y,z\colon (g(x,y) \wedge g(x,z) \rightarrow y = z)\ \},\end{aligned}$$

where the last two formulae are expressing that f, g are interpreted as partial functions, and $b_1, \ldots, c_3$ are Skolem constants. Note that these formulae have already been used to simplify the rest of the ABox, and that redundant equalities have been removed. We want to show that $b_3 = c_3$ is a consequence of the default theory.

The translated and Skolemized form of the consequent $fg \doteq gf$ of the third default is $f(x, h_1(x)) \wedge g(h_1(x), h_2(x)) \wedge g(x, k_1(x)) \wedge f(k_1(x), k_2(x)) \wedge h_2(x) = k_2(x)$, where h_1, h_2, k_1, k_2 are unary Skolem functions.

Since $A(a)$ is in our world description, the third default, instantiated by a, is applicable, and yields $f(a, h_1(a)) \wedge g(h_1(a), h_2(a)) \wedge g(a, k_1(a)) \wedge f(k_1(a), k_2(a)) \wedge h_2(a) = k_2(a)$. The formulae which express that f, g are partial functions yield $h_1(a) = b_1$, $h_2(a) = b_2$, and $k_1(a) = c_1$.

Applying the second default, instantiated by a, we get $\forall y\colon (g(a, y) \rightarrow A(y))$, which in turn yields $A(c_1)$. Now we can apply the third default, instantiated by c_1, which yields $f(c_1, h_1(c_1)) \wedge g(h_1(c_1), h_2(c_1)) \wedge g(c_1, k_1(c_1)) \wedge f(k_1(c_1), k_2(c_1)) \wedge h_2(c_1) = k_2(c_1)$. Because of the formulae expressing that f, g are partial functions we get $c_2 = k_1(c_1), c_3 = k_2(c_1)$, and, using the additional fact $k_1(a) = c_1$, also $k_2(a) = h_1(c_1)$.

To sum up we have $b_2 = h_2(a) = k_2(a) = h_1(c_1)$, $c_3 = k_2(c_1) = h_2(c_1)$, and $g(b_2, b_3)$ as well as $g(h_1(c_1), h_2(c_1))$. This yields $b_3 = h_2(c_1) = c_3$, which is what we wanted to show.

Since the word problem for semigroups is in general undecidable, the proposition shows that our terminological default theories in general have an undecidable consequence problem.

Corollary 4.2 *The consequence problem for an open default theory is in general undecidable, even if one has a finite set of defaults and the base language is decidable.*

It should be noted that the default rules used in the reduction are monotonic (i.e., they do not have justifications). Consequently, the default theory has ex-

actly one extension, which shows that the undecidability result is independent of whether one wants to employ skeptical or credulous reasoning. In addition, this shows that the consequences of rule applications in the CLASSIC system would become undecidable, if CLASSIC applied rules not only to individuals explicitly present in the ABox, but also to implicit individuals. This result for CLASSIC rules has already been mentioned by Nebel and Smolka [19], but without proof. In the next section we shall see that the restriction to explicit individuals leads to a decidable consequence relation even if one allows nonmonotonic default rules instead of CLASSIC's monotonic rules.

5 Computing Extensions

Because of the problems caused by Skolemization in Reiter's treatment of open defaults, we now propose a *restricted semantics for open default theories:* default rules are only applied to individuals that are explicitly mentioned in the ABox.

Definition 5.1 *In the restricted semantics for terminological default theories, an open default of a terminological default theory $(\mathcal{A}, \mathcal{D})$ is interpreted as representing the closed defaults obtained by instantiating the free variable by all individual names occurring in $\mathcal{A}$.*

Because the ABox $\mathcal{A}$ and the set of open defaults $\mathcal{D}$ are assumed to be finite, we end up with a *finite set of closed defaults.* Since our terminological language is decidable, the methods of Junker and Konolige, or of Schwind and Risch can be applied to compute all extensions (according to our restricted semantics).

In principle, both methods depend on the fact that any extension of a closed default theory $(\mathcal{A}, \mathcal{D})$ is of the form $Th(\mathcal{A} \cup Con(\widehat{\mathcal{D}}))$ for a subset $\widehat{\mathcal{D}}$ of $\mathcal{D}$. If $\mathcal{D}$ is finite, there are only finitely many such subsets, and the only problem is to decide which of these generate an extension. In fact, if the base language is decidable, one could even use for this purpose the iteration process described in the definition of an extension. This is so because decidability of the base language makes each iteration step effective, and the iteration process terminates because there are only finitely many consequents to be added. However, with this method one has to consider all the (exponentially many) subsets of $\mathcal{D}$. The two methods which we shall describe below try to avoid considering all subsets, thus making the search for (the sets of generating defaults of) all extensions more efficient.

5.1 Junker and Konolige's method

Junker and Konolige [10] translate a closed default theory $(\mathcal{A}, \mathcal{D})$ into a Truth Maintenance Network (TMN) à la Doyle [6]. The nodes of the TMN are the consequents $\mathcal{C}_\mathcal{D}$, and the prerequisites and negated justifications $\mathcal{L}_\mathcal{D}$ of the defaults. A default $\alpha : \beta_1, \ldots, \beta_n / \gamma$ of $\mathcal{D}$ is translated into a *nonmonotonic justification* $\langle in(\alpha), out(\neg\beta_1, \ldots, \neg\beta_n) \rightarrow \gamma \rangle$ of the TMN. In order to supply the truth maintenance system with enough information about first-order derivability in the base language, each prerequisite and negated justification of a default gives rise to several *monotonic justifications* of the TMN. These justifications are of the form $\langle in(Q) \rightarrow q \rangle$ where $q \in \mathcal{L}_\mathcal{D}$, and Q is a minimal subset of $\mathcal{C}_\mathcal{D}$ such that $\mathcal{A} \cup Q$ entails q—i.e., $\mathcal{A} \cup Q \models q$ but $\mathcal{A} \cup Q' \not\models q$ for every proper subset Q' of Q.

Junker and Konolige show that there is a 1–1-correspondence between admissible labellings of the TMN thus obtained and extensions of the default theory, and they describe an algorithm which computes all admissible labellings of a TMN. Given such an admissible labelling, the set of generating defaults of the corresponding extension consists of the defaults whose consequents are labelled "in."

In order to make the translation of terminological default theories into TMNs effective, one has to show how to compute the above mentioned monotonic justifications of the TMN. First note that the elements of $\mathcal{L}_\mathcal{D} \cup \mathcal{C}_\mathcal{D}$ are admissible assertional facts. This is obvious for the prerequisites and the consequents of our instantiated defaults, and for the negated justifications it follows from the fact that the concept language has negation as an operator. For this reason, $\mathcal{A} \cup Q$ for a subset Q of $\mathcal{C}_\mathcal{D}$ is an admissible ABox of our language, and the entailment problem $\mathcal{A} \cup Q \models q$ for $q \in \mathcal{L}_\mathcal{D}$ is an ordinary instantiation problem. As mentioned in Section 2, the instantiation problem is decidable for our language. A *brute force algorithm* could just compute all subsets Q of $\mathcal{C}_\mathcal{D}$ such that $\mathcal{A} \cup Q$ entails $q \in \mathcal{L}_\mathcal{D}$, and then, for each q, eliminate the ones which are not minimal. Of course, this simple algorithm is very inefficient, and thus not appropriate for actual implementations.

Because $\mathcal{A} \cup Q$ entails an assertional fact $C(a)$ iff $\mathcal{A} \cup Q \cup \{\neg C(a)\}$ is inconsistent, we need a solution of the following problem: Let $\mathcal{A}$, $\mathcal{B}$ be ABoxes. Find all minimal subsets $\mathcal{Q}$ of $\mathcal{B}$ such that $\mathcal{A} \cup \mathcal{Q}$ is inconsistent. Since a similar algorithmic problem has to be solved for the method obtained from Schwind and Risch's characterization of an extension, we defer the description of a more efficient solution of this problem to a separate section.

A characteristic feature of Junker and Konolige's method is that—after the computation of the minimal sets Q—it is completely abstracted from derivability in the base language. This may be advantageous from a conceptual point of view, but it can be problematic from the algorithmic point of view. In fact, one has to compute the corresponding minimal sets for all elements q in $\mathcal{L}_\mathcal{D}$, even though this information may not contribute to the computation of an extension.

5.2 A method based on a theorem by Schwind and Risch

Schwind and Risch [25] give a theorem which characterizes those subsets $\widehat{\mathcal{D}}$ of $\mathcal{D}$ which are sets of generating defaults of an extension of a closed default theory $(\mathcal{W}, \mathcal{D})$. They use this characterization for computing extensions of propositional default theories. In this subsection, we shall show how to apply the theorem to computing extensions of terminological default theories.

Before we can formulate the theorem we need one more piece of notation.

Definition 5.2 *Let $\mathcal{W}$ be a set of closed formulae, and $\mathcal{D}$ be a set of closed defaults. We define $\mathcal{D}_0 = \emptyset$ and, for $i \geq 0$,*

$$\mathcal{D}_{i+1} = \mathcal{D}_i \cup \{d = \alpha : \beta_1, \ldots, \beta_n/\gamma \mid d \in \mathcal{D} \text{ and } \mathcal{W} \cup Con(\mathcal{D}_i) \models \alpha\}.$$

Then $\mathcal{D}$ is called grounded *in $\mathcal{W}$ iff $\mathcal{D} = \bigcup_{i=0}^{\infty} \mathcal{D}_i$.*

This definition of groundedness differs from the one given in [25], but it is easy to see that both formulations are equivalent. The advantage of our formulation is that it can directly be used as a procedure for deciding groundedness, if $\mathcal{D}$ is finite and the entailment problem in the base language is decidable. If $\mathcal{D}$ is not grounded in $\mathcal{W}$, then $\bigcup_{i=0}^{\infty} \mathcal{D}_i$ is the largest subset of $\mathcal{D}$ that is grounded in $\mathcal{W}$.

The iteration process described above corresponds to the iteration in the definition of extensions, with the main difference that it disregards the justifications. The second condition given in the following theorem makes up for this neglect.

Theorem 5.3 (Schwind and Risch) *Let $(\mathcal{W}, \mathcal{D})$ be a closed default theory. A subset $\widehat{\mathcal{D}}$ of $\mathcal{D}$ is a set of generating defaults of an extension of $(\mathcal{W}, \mathcal{D})$ iff the following two conditions hold:*

1. *$\widehat{\mathcal{D}}$ is grounded in $\mathcal{W}$.*
2. *For all $d \in \mathcal{D}$ with $d = \alpha : \beta_1, \ldots, \beta_n/\gamma$ we have $d \in \widehat{\mathcal{D}}$ iff $\mathcal{W} \cup Con(\widehat{\mathcal{D}}) \models \alpha$ and for all $i, 1 \leq i \leq n$, $\mathcal{W} \cup Con(\widehat{\mathcal{D}}) \not\models \neg\beta_i$.*

If $\mathcal{D}$ is finite, and the entailment problem in the base language is decidable, this theorem provides us with an effective test of whether a subset $\widehat{\mathcal{D}}$ of $\mathcal{D}$ is a set of generating defaults of an extension of $(\mathcal{W}, \mathcal{D})$. We shall now describe a method based on this theorem which allows us to compute (the sets of generating defaults of) all extensions without having to consider all subsets of $\mathcal{D}$.

If $\mathcal{W}$ is inconsistent then there is only one extension, namely the set of all formulae. In the following, we shall without loss of generality assume that $\mathcal{W}$ is consistent. Now, let $\mathcal{D}_1, \ldots, \mathcal{D}_m$ be all maximal subsets of $\mathcal{D}$ such that $\mathcal{W} \cup Con(\mathcal{D}_i)$ is consistent. Since $\mathcal{W}$ is assumed to be consistent, extensions are consistent as well, which means that a generating set of defaults of an extension is a subset of one of the $\mathcal{D}_i$. The idea underlying our method is to start with these maximal sets $\mathcal{D}_i$, and successively eliminate defaults violating the first condition of the theorem, or the "only if" part of the second condition. If no more defaults can be eliminated, the "if" part of the second condition is tested.

Figure 1 describes the procedure for computing all extensions of a closed default theory. This procedure uses the following subprocedures which have not explicitly been described:

- Decide whether $\mathcal{W}$ is consistent.
- Compute all maximal subsets $\mathcal{D}'$ of $\mathcal{D}$ such that $\mathcal{W} \cup Con(\mathcal{D}')$ is consistent.
- Compute the largest subset $\mathcal{D}_0$ of $\mathcal{D}'$ that is grounded in $\mathcal{W}$.
- Compute all maximal subsets $\mathcal{D}''$ of $\mathcal{D}_0$ such that $\mathcal{W} \cup Con(\mathcal{D}'') \not\models \neg\beta_i$.

The first subprocedure is a direct application of the decision algorithm for entailment in the base language. The third subprocedure is simply obtained by implementing the definition of groundedness.

The other two procedures depend on an algorithm for the following problem, which will be considered in the next section: Let $\mathcal{A}, \mathcal{B}$ be ABoxes. Compute all maximal subsets $\mathcal{Q}$ of $\mathcal{B}$ such that $\mathcal{A} \cup \mathcal{Q}$ is consistent.

In fact, the second subprocedure is a direct application of such an algorithm. For the fourth subprocedure, note that $\mathcal{W} \cup Con(\mathcal{D}'') \not\models \neg\beta_i$ iff $\mathcal{W} \cup Con(\mathcal{D}'') \cup \{\beta_i\}$ is consistent.

6 Computing minimal inconsistent and maximal consistent ABoxes

This section is concerned with the following algorithmic problems: Given two ABoxes $\mathcal{A}, \mathcal{B}$, find all minimal (resp. maximal) subsets $\mathcal{Q}$ of $\mathcal{B}$ such that $\mathcal{A} \cup \mathcal{Q}$ is inconsistent (resp. consistent).

Since consistency of ABoxes in our language is decidable, there is the obvious "brute-force" solution which tests consistency of $\mathcal{A} \cup \mathcal{Q}$ for all subsets $\mathcal{Q}$ of $\mathcal{B}$, and then takes the minimal inconsistent (maximal consistent) ones. In the following we shall describe a more efficient method of finding these minimal (maximal) sets. The method is an extension of the tableaux-based consistency algorithms for ABoxes described in [1, 8]. The idea of employing tableaux-based methods for such purposes was already used in [16, 25], but they

```
Compute-All-Extensions(W, D)
begin
    if W is inconsistent
      then print "Inconsistent world description"
      else for all maximal subsets D' of D such that W ∪ Con(D') is consistent
              Remove-Defaults(W, D, D');
end

Remove-Defaults(W, D, D')
begin
    let D_0 be the largest subset of D' that is grounded in W;
    if W ∪ Con(D_0) ⊨ ¬β_i for some justification β_i ∈ Jus(D_0)
      then let d = α : β_1, ..., β_n/γ be the corresponding default;
           Remove-Defaults(W, D, D_0 \ {d});
           for all maximal subsets D'' of D_0 such that d ∈ D'' and W ∪ Con(D'') ⊭ ¬β_i
              Remove-Defaults(W, D, D'');
      else if for each α : β_1, ..., β_n/γ ∈ D \ D_0 either A ∪ Con(D_0) ⊭ α or A ∪ Con(D_0) ⊨ ¬β_i for some i
             then add D_0 to the list of generating sets of defaults;
end
```

Figure 1: Procedure for computing the generating sets of defaults of all extensions of the closed default theory $(\mathcal{W}, \mathcal{D})$. *Proviso:* $\mathcal{D}$ is finite and entailment in the base language is decidable.

restricted themselves to propositional logic, which is a much easier case.

In order to decide whether an ABox $\mathcal{A}$ is consistent, the tableaux-based consistency algorithm tries to generate a finite model of $\mathcal{A}$. In principle, it starts with $\mathcal{A}$, and adds new assertional facts with the help of certain rules until the obtained ABox is "complete," i.e., one can apply no more rules. Because of the presence of disjunction in our language, a given ABox must sometimes be transformed into two different new ABoxes, with the intended meaning that the original ABox is consistent iff one of the new ABoxes is consistent. Formally, this means that one is working with sets of ABoxes instead of a single ABox. It can be shown that the transformation process always terminates, and that $\mathcal{A}$ is inconsistent iff each of the finitely many complete ABoxes derived from $\mathcal{A}$ are "obviously contradictory."

Now assume that $\mathcal{A}, \mathcal{B}$ are ABoxes, and we want to find all minimal (resp. maximal) subsets $\mathcal{Q}$ of $\mathcal{B}$ such that $\mathcal{A} \cup \mathcal{Q}$ is inconsistent (resp. consistent). We start with applying the tableaux-based consistency algorithm to $\mathcal{A} \cup \mathcal{B}$. Let $\mathcal{A}_1, \ldots, \mathcal{A}_m$ be the complete ABoxes obtained this way. If one of these is not obviously contradictory, $\mathcal{A} \cup \mathcal{B}$ is consistent, and there are no minimal inconsistent sets to compute (resp. $\mathcal{B}$ is the maximal consistent set). Otherwise, we want to know which elements of $\mathcal{B}$ can be dispensed with without destroying the property that all complete ABoxes contain an obvious contradiction (resp. which elements of $\mathcal{B}$ have to be removed to get at least one complete ABox without obvious contradiction).

For this reason, it is important to know which facts in $\mathcal{B}$ contribute to a particular obvious contradiction. To this purpose we introduce a propositional variable for each element of $\mathcal{B}$, and label assertional facts with "monotonic" boolean formulae built from these variables, i.e., propositional formulae built from the variables by using conjunction and disjunction only. In the original ABox $\mathcal{A} \cup \mathcal{B}$, the elements of $\mathcal{A}$ are labelled with "true," and the elements of $\mathcal{B}$ are labelled with the corresponding propositional variable. If, during the consistency test, n assertional facts with labels $\phi_1, \ldots, \phi_n$ give rise to a new fact, the new one is labelled by $\phi_1 \wedge \ldots \wedge \phi_n$. Since the same assertional fact may arise in more than one way, we also get disjunctions in labels. Again, we end up with complete ABoxes $\mathcal{A}_1, \ldots, \mathcal{A}_m$, but now all assertional facts occurring in these ABoxes have labels.

More formally, we shall now describe a ***labelled consistency algorithm*** for ABoxes $\mathcal{A} \cup \mathcal{B}$ consisting of "hard" facts $\mathcal{A}$ and of "refutable" facts $\mathcal{B}$. For ease of presentation, we restrict ourselves in this formal description to the terminological language $\mathcal{ALC}$ where we do not have attributes and agreements. Later on, we shall point out how the algorithm can be extended to $\mathcal{ALCF}$.

Without loss of generality we assume that the concept terms occurring in $\mathcal{A} \cup \mathcal{B}$ are in negation normal form, i.e., negation occurs only directly in front of concept names. Initially, the elements of $\mathcal{A} \cup \mathcal{B}$ are labelled with monotonic boolean formulae as described above. We shall refer to the label of an assertional fact α by $ind(\alpha)$. Starting with the singleton set $\{\mathcal{A} \cup \mathcal{B}\}$, the transformation rules of Figure 2 are applied as long as

Let $\mathcal{M}$ be a finite set of labelled ABoxes, and let $\mathcal{A}_0$ be an element of $\mathcal{M}$. The following rules will replace $\mathcal{A}_0$ by an ABox $\mathcal{A}_1$ or by two ABoxes $\mathcal{A}_1$ and $\mathcal{A}_2$. These new ABoxes either contain additional assertional facts, or the indices of existing assertional facts are changed. In order to avoid having to distinguish between these two cases in the formulation of the rules, we introduce a new notation. An ABox is *extended by* an assertional fact with index ϕ means the following: If this fact is already present with index ψ, we just change its index to $\psi \vee \phi$. Otherwise, it is added to the ABox and gets index ϕ.

The conjunction rule. Assume that $(C \sqcap D)(a)$ is in $\mathcal{A}_0$, and that $\mathcal{A}_0$ does not contain assertions $C(a)$ and $D(a)$ whose indices are both implied by $ind((C \sqcap D)(a))$. The ABox $\mathcal{A}_1$ is obtained by extending $\mathcal{A}_0$ by $C(a)$ with index $ind((C \sqcap D)(a))$ and by $D(a)$ with index $ind((C \sqcap D)(a))$.

The disjunction rule. Assume that $(C \sqcup D)(a)$ is in $\mathcal{A}_0$, and that $\mathcal{A}_0$ does not contain $C(a)$ or $D(a)$ whose index is implied by $ind((C \sqcup D)(a))$. The ABox $\mathcal{A}_1$ is obtained by extending $\mathcal{A}_0$ by $C(a)$ with index $ind((C \sqcup D)(a))$, and the ABox $\mathcal{A}_2$ is obtained by extending $\mathcal{A}_0$ by $D(a)$ with index $ind((C \sqcup D)(a))$.

The exists-restriction rule. Assume that $(\exists R.C)(a)$ is in $\mathcal{A}_0$, and that $\mathcal{A}_0$ does not contain assertions $R(a,b)$ and $C(b)$ whose indices are both implied by $ind((\exists R.C)(a))$. If $\mathcal{A}_0$ contains assertions $R(a,b)$ and $C(b)$ for some individual b then $\mathcal{A}_1$ is obtained by extending $\mathcal{A}_0$ by $R(a,b)$ with index $ind((\exists R.C)(a))$ and by $C(b)$ with index $ind((\exists R.C)(a))$. Otherwise, one generates a new individual name b, and obtains $\mathcal{A}_1$ from $\mathcal{A}_0$ by adding $R(a,b)$ and $C(b)$, both with index $ind((\exists R.C)(a))$.

The value-restriction rule. Assume that $(\forall R.C)(a)$ and $R(a,b)$ are in $\mathcal{A}_0$, and that $\mathcal{A}_0$ does not contain an assertion $C(b)$ whose index is implied by $ind((\forall R.C)(a)) \wedge ind(R(a,b))$. The ABox $\mathcal{A}_1$ is obtained by extending $\mathcal{A}_0$ by $C(b)$ with index $ind((\forall R.C)(a)) \wedge ind(R(a,b))$.

Figure 2: Transformation rules of the labelled consistency algorithm for $\mathcal{ALC}$.

possible. Since there cannot be an infinite chain of rule applications (see [2] for a proof), we finally end up with a finite set of complete ABoxes, i.e., ABoxes to whom no more rules apply. As for the unlabelled consistency algorithm, the original ABox is inconsistent iff all these complete ABoxes contain an "obvious" contradiction, called clash. For $\mathcal{ALC}$, the only clash case is the presence of an assertional fact $A(a)$ and its negation $\neg A(a)$ for an individual name a and a concept name A.

The labels occurring in the complete ABoxes have the following important property.

Proposition 6.1 *Let $\mathcal{Q} \subseteq \mathcal{B}$, and consider the valuation which replaces the propositional variables corresponding to elements of $\mathcal{Q}$ by "true" and the others by "false." If we remove from the complete ABoxes obtained by applying the labelled consistency algorithm to $\mathcal{A} \cup \mathcal{B}$ all facts whose labels evaluate to "false" then we end up with the complete ABoxes which would be obtained when starting an unlabelled consistency algorithm with $\mathcal{A} \cup \mathcal{Q}$.*

Using the labels, whether after such a removal each of the remaining complete ABoxes still contains an obvious contradiction can easily be checked by considering the following monotonic boolean formula.

Definition 6.2 (Clash formula) *Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be the complete ABoxes obtained by applying the labelled consistency algorithm to $\mathcal{A} \cup \mathcal{B}$. A particular clash $A(a), \neg A(a) \in \mathcal{A}_i$ is expressed by the propositional formula $ind(A(a)) \wedge ind(\neg A(a))$. Now let $\psi_{i,1}, \ldots, \psi_{i,k_i}$ be the formulae expressing all the clashes in $\mathcal{A}_i$. The clash formula associated with $\mathcal{A} \cup \mathcal{B}$ is*

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} \psi_{i,j}.$$

We have used conjunction when expressing a single clash because both assertional facts are necessary for the contradiction. Now recall that we need *at least one* clash *in each* of the complete ABoxes to have inconsistency. This explains why disjunction is used to combine the formulae expressing the clashes of one complete ABox, and why the formulae corresponding to the different complete ABoxes are combined with the help of conjunction.

Proposition 6.3 *Let ψ be the clash formula associated with $\mathcal{A} \cup \mathcal{B}$, let $\mathcal{Q} \subseteq \mathcal{B}$, and let ω be the valuation which replaces the propositional variables corresponding to elements of $\mathcal{Q}$ by "true" and the others by "false." Then $\mathcal{A} \cup \mathcal{Q}$ is inconsistent iff ψ evaluates to "true" under ω.*

Thus the minimal (resp. maximal) subsets $\mathcal{Q}$ of $\mathcal{B}$ such that $\mathcal{A} \cup \mathcal{Q}$ is inconsistent (resp. consistent) correspond to minimal (resp. maximal) valuations making the clash formula ψ "true" (resp. "false"). Here "minimal" and "maximal" for valuations is meant with respect

to the partial ordering $\omega_1 \leq \omega_2$ iff $\omega_1(p_i) \leq \omega_2(p_i)$ for all propositional variables p_i, where we assume that "false" is smaller than "true."

It is easy to see that the problem of finding maximal valuations making a monotonic boolean formula "false" can be reduced to the problem of finding minimal valuations making a monotonic boolean formula "true." In fact, for a given monotonic boolean formula ψ and a valuation ω, let ψ^d denote the formula obtained from ψ by replacing conjunction by disjunction and vice versa, and let ω^d denote the valuation obtained from ω by replacing "true" by "false" and vice versa. Then ω is a maximal valuation making ψ "false" iff ω^d is a minimal valuation making ψ^d "true."

It should be noted that the problem of finding minimal valuations of monotonic boolean formulae is NP-complete. In fact, if ψ is in conjunctive normal form, this is just the well-known problem of finding minimal hitting sets [24, 7]. On the other hand, if ψ is in disjunctive normal form, the minimal valuations can be found in polynomial time. However, transforming a given monotonic boolean formula into disjunctive normal form may cause an exponential blow-up.

The rules of the labelled consistency algorithm as described have the unpleasant property that deciding whether or not a rule is applicable is an NP-hard problem. In fact, the preconditions of the rules include an entailment test for monotonic boolean formulae, which is NP-hard. However, one can weaken the precondition by testing a necessary condition for entailment (e.g. occurrence of the index in the top-level disjunction) without destroying the property stated in Proposition 6.1. In this case, the rules will in general produce longer formulae occurring as indices, but the test whether a rule applies becomes tractable.

In the remaining part of this section we shall sketch how the above described algorithm can be extended to handle the attributes and agreements of $\mathcal{ALCF}$.

Attributes in exists- and value-restrictions are treated like roles. Applying the exists-rule to two assertional facts $(\exists f.C)(a)$ and $(\exists f.D)(a)$ introduces two different individual names c, d with the assertional facts $f(a,c), f(a,d)$. If f is an attribute, this means that c and d have to be interpreted as the same individual. This shows that we can no longer have a unique name assumption for the individuals which are introduced by rules. For this reason, we shall now distinguish between "old" individuals, i.e., individuals present in the original ABox $\mathcal{A} \cup \mathcal{B}$, and "new" individuals introduced by rule applications. New individuals are not subjected to the unique name assumption. In order to make the constraint that c, d have to be interpreted by the same individual explicit, the consistency algorithm for $\mathcal{ALCF}$ (see [9]) identifies these two individual names, e.g., by replacing every occurrence of c by d. In the labelled consistency algorithm, instead of making an actual replacement, we just introduce an equality fact $c = d$. Of course, this equality has to be equipped with an index, in the same way as other facts are. Here the fact $c = d$ gets index $ind(f(a,c)) \wedge ind(f(a,d))$ if it is newly introduced, otherwise one takes the disjunction of its old index with $ind(f(a,c)) \wedge ind(f(a,d))$. In case $ind(f(a,c)) \wedge ind(f(a,d))$ implies the old index, nothing has to be changed.

With the help of the equality facts, it is easy to formulate an *agreement rule*. In principle, the agreement rule applied to $(f_1 \cdots f_m \doteq g_1 \cdots g_n)(a)$ introduces the assertional facts $f_1(a,c_1), \ldots, f_m(c_{m-1}, c_m)$, $g_1(a,d_1), \ldots, g_n(d_{n-1}, d_n)$ and $c_m = d_n$, where $c_1, \ldots, d_n$ are new individual names. Applicability of this rule, and the indices of the new facts (or new indices of existing facts) are defined analogously to the other rules.

The equality facts define an equivalence relation on individual names, which has to be taken into account when firing rules or looking for clashes. Premises of rules have to be read modulo this equivalence. For example, this means that the value-restriction rule may be applicable to the facts $(\forall R.C)(a)$ and $R(a', b)$, if there are equalities $a = a_0, a_1 = a_2, \ldots, a_n = a'$ in the ABox. Of course, the indices of these equalities have to contribute to the new index of $C(b)$ as well. On the other hand, this rule need not be applied if there exists an assertional fact $C(b')$ and equalities $b = b_0, b_1 = b_2, \ldots, b_m = b'$ such that $ind((\forall R.C)(a)) \wedge ind(R(a', b)) \wedge ind(a = a_0) \ldots \wedge ind(a_n = a')$ implies $ind(C(b')) \wedge ind(b = b_0) \ldots \wedge ind(b_m = b')$.

Similarly, there is a clash if $A(a)$ and $\neg A(a')$ is in the ABox, along with equalities $a = a_0, a_1 = a_2, \ldots, a_n = a'$. Because we still have unique name assumption for the old individuals, the equalities may cause another kind of obvious contradiction. We have a clash if a, a' are old individuals and there are equalities $a = a_0, a_1 = a_2, \ldots, a_n = a'$ in the ABox. The index associated with this clash is $ind(a = a_0) \wedge \ldots \wedge ind(a_n = a')$.

To sum up, we thus have a solution of the two algorithmic problems described at the beginning of this section. Together with the methods of Section 5 this give us effective procedures to compute all extensions of terminological default theories.

7 Conclusion

We have investigated the integration of Reiter's default logic into a terminological representation formalism, and have shown that the treatment of open defaults by Skolemization is problematic, both from a semantic and an algorithmic point of view. For this reason, we have considered a restricted semantics where default rules are only applied to individuals explicitly present in the knowledge base. This treatment of default rules is similar to the treatment of monotonic rules in many

terminological systems, which means that users of such systems are already familiar with the effects this restriction to explicit individuals has. However, because of the nonmonotonic character of default rules, this restriction may sometimes lead to more consequences than would have been obtained without it.

With respect to the restricted semantics, the methods of Junker and Konolige and of Schwind and Risch for computing all extensions of a default theory can be applied. We have shown how the algorithmic requirements for Junker and Konolige's method (i.e., the computation of minimal inconsistent sets of assertional facts) and for an optimized algorithm based on a theorem of Schwind and Risch (i.e., the computation of maximal consistent sets of assertional facts) can be solved by an extension of the tableaux-based algorithm for assertional reasoning.

According to Reiter's semantics the specificity of prerequisites of rules has no influence on the order in which defaults rules are supposed to fire. To realize this additional feature one could either consider an appropriate selection of admissible extensions, or use prioritized default theories.

Acknowledgements

We should like to thank Bernhard Nebel and Peter Patel-Schneider for helpful comments.

This work has been supported by the German Ministry for Research and Technology (BMFT) under research contract ITW 8903 0.

References

[1] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of IJCAI'91*, Sydney, Australia, 1991.

[2] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. Research Report, DFKI Saarbrücken, 1992.

[3] R. J. Brachman. 'I lied about the trees' or, defaults and definitions in knowledge representation. *The AI Magazine*, 6(3):80–93, 1985.

[4] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In J. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, San Mateo, Calif., 1991.

[5] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[6] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[7] M. Garey and D. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, Cal., 1979.

[8] B. Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proceedings of GWAI'90*, Ebingerfeld, Germany, 1990.

[9] B. Hollunder and W. Nutt. Subsumption Algorithms for Concept Languages. Research Report RR-90-04, DFKI Kaiserslautern, 1990.

[10] U. Junker and K. Konolige. Computing extensions of autoepistemic and default logics with a truth maintenance system. In *Proceedings AAAI'90*, pages 278–283, Boston, Ma., 1990.

[11] H. A. Kautz and B. Selman. Hard problems for simple defaults. In *Proceedings of KR'89*, pages 189–197, Toronto, Ont., 1989.

[12] A. Kobsa. The SB-ONE knowledge representation workbench. In *Preprints of the Workshop on Formal Aspects of Semantic Networks*, Two Habours, Calif., 1989.

[13] V. Lifschitz. On open defaults. In *Proceedings of the Symposium on Computational Logics*, Brüssel, Belgium, 1990.

[14] E. Mays and B. Dionne. Making KR systems useful. In [20], pages 11–12.

[15] J. McCarthy. Circumscription – a form of nonmonotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[16] D. McDermott and J. Doyle. Non-monotonic logic I. *Artificial Intelligence*, 13:41–72, 1980.

[17] R. McGregor. Statement of interest. In K. von Luck, B. Nebel, and C. Peltason, editors, *Statement of Interest for the 2nd International Workshop on Terminological Logics.* Document D-91-13, DFKI Kaiserslautern, 1991.

[18] μBACK. System presentation. In [20], page 186.

[19] B. Nebel and G. Smolka. Attributive description formalisms ... and the rest of the world. In C. Rollinger O. Herzog, editor, *Text Understanding in LILOG*, LNAI 546. Springer-Verlag, Berlin, Germany, 1991.

[20] C. Peltason, K. v. Luck, and C. Kindermann (Org.). Terminological logic users workshop – Proceedings. KIT Report 95, TU Berlin, 1991.

[21] D. L. Poole. Variables in hypothesis. In *IJCAI'87*, Milano, Italy, 1987.

[22] E. L. Post. Recursive unsolvability of a problem of Thue. *Journal of Symbolic Logic*, 12:1–10, 1947.

[23] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[24] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[25] C. Schwind and V. Risch. A tableau-based characterisation for default logic. In *Proceedings of the 1st European Conference on Symbolic and Quantitative Approaches for Uncertainty*, pages 310–317, Marseilles, France, 1991.

Specifying Role Interaction in Concept Languages

Philipp Hanschke
German Research Center for AI (DFKI)
Postfach 2080
W-6750 Kaiserslautern
Germany

Abstract

The KL-ONE concept language provides role-value maps (RVMs) as a concept forming operator that compares *sets* of role fillers. This is a useful means to specify structural properties of concepts. Recently, it has been shown that concept languages providing RVMs together with some other common concept-forming operators induce an undecidable subsumption problem. Thus, RVMs have been restricted to chainings of *functional* roles as, for example, in CLASSIC. Although this restricted RVM is still a useful operator, one would like to have additional means to specify interaction of general roles. The present paper investigates two concept languages for that purpose. The first one provides concept forming operators that generalize the restricted RVM in a different direction. Unfortunately, it turns out that this language also has an undecidable subsumption problem. The second formalism allows to specify structural properties w.r.t. roles without using general equality and is equipped with (complete) decision procedures for its associated reasoning problems.

1 INTRODUCTION

Concept languages based on KL-ONE [Brachman and Schmolze, 1985] are mostly used to represent the terminological knowledge of a particular problem domain on an abstract logical level. To describe this kind of knowledge, one starts with atomic concepts and roles, and defines new concepts using the operations provided by the language. Concepts can be considered as unary predicates which are interpreted as sets of individuals, and roles as binary predicates which are interpreted as binary relations between individuals. Examples for atomic concepts may be Human and Female, and for roles friend and enemy. Many terminological formalisms concentrate on the following three categories of operators to build a terminology:

- *Boolean connectives* ($\sqcap$, $\sqcup$, and $\neg$) that allow concepts to be combined without any direct reference to their internal structure. For example, if the logical connective conjunction is present as a language construct, one may describe the concept Woman as "humans who are female", and represent it by the expression Human $\sqcap$ Female.
- *Role-forming operators* that allow new roles to be defined. For example the composition ($\circ$) allows the role "friend of enemy" to be represented by enemy $\circ$ friend.
- Operators on *role fillers* that allow the 'internal' structure of the concepts to be operated on. Many languages provide quantification over role fillers which allows, for example, the concept "human with a friend" (resp., "human with only female friends") to be described by the expression Human $\sqcap \exists$friend.Human (resp., Human $\sqcap$ $\forall$friend.Female). An interesting subclass of operators on role fillers are the operators for *role interaction*. The frequently used *number restrictions* can be seen as a degenerated form to specify role interaction (on one role). For example, the concept Lucky-Human could be defined as $\exists^{>100}$friend $\sqcap \exists^{<2}$enemy. As soon as an individual belonging to this concept has two role fillers for enemy, it can be deduced that they are equal.

The kind of models that can be specified by the operators considered so far is quite restricted. If a concept C is satisfiable, then it is satisfiable by an interpretation that arranges its individuals in a tree structure. For example, it is possible to require that the members of a concept have role fillers for a role R, say an individual a, and a role S, say b. But it is not possible to specify that a equals b or that a and b have any common (transitive) role-filler, or that their respective role-fillers are in any relation to each other.

So there is a need for additional means to specify role interaction. The classical prototypes of this kind

of operators are the *structural descriptions* and *role-value maps* (RVMs; see Section 3 for a definition) that are discussed and motivated, e.g., in [Brachman and Schmolze, 1985].

An RVM would allow one to specify that the set of all friends of an individual is equal to the set of all enemies (which may be true for some people if one looks at some never ending soap operas): enemy $=_{\mathrm{RVM}}$ friend where enemy and friend are roles.

In [Schmidt-Schauß, 1989; Patel-Schneider, 1989] it has been shown that a concept language with RVMs and a few other common operators has an undecidable subsumption problem. As a reaction on this disappointing negative result, RVMs have been restricted in existing systems to attribute agreements, see, e.g., [Borgida *et al.*, 1989]. *Attributes* are functional roles and are sometimes also called features. I.e., they have at most one role filler per object. Let best-friend and main-enemy be attributes. Then an individual belongs to the concept main-enemy $=_{\mathrm{RVM}}$ best-friend if it does not have a main enemy, or if it does not have a best friend, or if its best friend is at the same time its main enemy.[1]

In this paper several other operators for specifying interaction of role and attribute fillers are investigated. The *existential role/attribute agreement* can be used to specify that there is at least one enemy that is also a friend: $\exists$(enemy = friend). If this operator is restricted to attribute chainings it is just the same-as operator in CLASSIC.

The expression $\exists$(enemy $\circ$ best-friend = friend $\circ$ best-friend) represents that there is at least one enemy and one friend who have the same best-friend. The *universal* agreement is used in the expression $\forall$(enemy $\circ$ best-friend = friend $\circ$ best-friend) to formalize that the best-friends of all friends and enemies are the same. On attribute chainings this construct agrees with the RVMs.

The *existential role/attribute disagreement* can express that there is at least one enemy and one friend that are not identical: $\exists$(enemy $\neq$ friend). The expression $\forall$(enemy $\neq$ friend) says that each member has only true friends and true enemies—there is no filler that is both a friend and an enemy.

Although it is at least not obvious how RVMs (on roles) can be simulated by this group of operators, it turns out that the existential and universal agreements lead to an undecidable subsumption problem (Section 3), too.

Section 4 introduces a new concept language which is able to relate fillers of role/attribute chainings. The main idea is to replace the general "=" (resp., "$\neq$") above, by abstract, not further defined predicates or by predicates of a *concrete domain*. In [Baader and Hanschke, 1991a] we already proposed an extension scheme with concrete domains, but there, the predicates are only applied to chainings of attributes. The present paper generalizes this extension scheme considerably.

As an example, consider the classic (toy) domain of families. Let age, spouse, and husband be attributes, child a role, and Male, Human not further defined concepts. Then a family could be represented by

$$\begin{array}{lcl} \mathsf{Woman} & = & \mathsf{Human} \sqcap \mathsf{Female} \\ \mathsf{Man} & = & \mathsf{Human} \sqcap \neg\mathsf{Female} \\ \mathsf{Family} & = & \exists \mathsf{husband.man} \sqcap \exists \mathsf{spouse.woman} \sqcap \\ & & \forall \mathsf{child.human} \end{array}$$

The specification can be further refined by enforcing that there is a marriage certificate and that children are younger than their parents.

$$\begin{array}{l} \mathsf{Normal\text{-}family} = \mathsf{Family} \sqcap \\ \qquad \forall(\mathsf{child} \circ \mathsf{age} > \mathsf{spouse} \circ \mathsf{age}) \sqcap \\ \qquad \exists \mathsf{husband}, \mathsf{spouse.marriage\text{-}certificate} \end{array}$$

Here the concrete predicate ">" and an abstract binary predicate marriage-certificate are used to formulate the additional requirements.

Section 5 sketches sound and complete reasoning algorithms (see Section 6 for a proof) for this expressive concept language with attribute (dis)agreements and the new structural description operators that are based on predicates.

The concept language $\mathcal{ALCF}$ is the basis for the two extensions and is defined in the next section.

2 THE BAISC LANGUAGE

This section introduces the language $\mathcal{ALCF}$ as a prototypical conventional concept language. It will be the starting point for the extensions described in the following sections.

Definition 2.1 (T-box syntax) Concept terms *are built from concept, role, and attribute names using concept-forming operators. If C and D are syntactic variables for concept terms and R is a role or attribute name, then*

$$\begin{array}{ll} C \sqcap D & \text{(conjunction)}, \\ C \sqcup D & \text{(disjunction)}, \\ \neg C & \text{(negation)} \\ \exists R.C & \text{(exists-in restriction)}, \textit{ and} \\ \forall R.C & \text{(value restriction)} \end{array}$$

are concept terms.

Let A be a concept name and let D be a concept term. Then $A = D$ is a terminological axiom. *A* terminology (T-box) *is a finite set $\mathcal{T}$ of terminological axioms*

[1] Actually, in CLASSIC the same-as operator requires the existence of one main-enemy and one best-friend.

with the additional restrictions that no concept name appears more than once as a left hand side of a definition, and $\mathcal{T}$ contains no cyclic definitions.[2]

A concept name that does not occur on the left side of a concept definition is called primitive. □

Please note that the exists-in and the value restrictions are not only defined for roles but also for attributes. The next definition gives a model-theoretic semantics for the language introduced in Definition 2.1.

Definition 2.2 (T-box semantics) *An* interpretation $\mathcal{I}$ *for $\mathcal{ALCF}$ consists of a set $dom(\mathcal{I})$ and an interpretation function. The interpretation function associates with each concept name A a subset $A^{\mathcal{I}}$ of $dom(\mathcal{I})$, with each role name R a binary relation $R^{\mathcal{I}}$ on $dom(\mathcal{I})$, i.e., a subset of $dom(\mathcal{I}) \times dom(\mathcal{I})$, and with each attribute name f a partial function $f^{\mathcal{I}}$ from $dom(\mathcal{I})$ into $dom(\mathcal{I})$. For such a partial function $f^{\mathcal{I}}$ the expression $f^{\mathcal{I}}(x) = y$ is sometimes written as $(x, y) \in f^{\mathcal{I}}$.*

The interpretation function—which gives an interpretation for atomic terms—can be extended to arbitrary concept terms as follows: Let C and D be concept terms and let R be a role or attribute name. Assume that $C^{\mathcal{I}}$ and $D^{\mathcal{I}}$ are already defined. Then

1. $a \in (C \sqcup D)^{\mathcal{I}}$ *iff* $a \in C^{\mathcal{I}}$ *or* $a \in D^{\mathcal{I}}$,
   $a \in (C \sqcap D)^{\mathcal{I}}$ *iff* $a \in C^{\mathcal{I}}$ *and* $a \in D^{\mathcal{I}}$,
   $a \in (\neg C)^{\mathcal{I}}$ *iff* $a \in dom(\mathcal{I}) \setminus C^{\mathcal{I}}$,
2. $a \in (\forall R.C)^{\mathcal{I}}$ *iff*
   for all y *with* $(x, y) \in R^{\mathcal{I}}$ *we have* $y \in C^{\mathcal{I}}$, *and*
   $a \in (\exists R.C)^{\mathcal{I}}$ *iff*
   there exists y *with* $(x, y) \in R^{\mathcal{I}}$ *and* $y \in C^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ is a model *of the T-box $\mathcal{T}$ iff it satisfies $A^{\mathcal{I}} = D^{\mathcal{I}}$ for all terminological axioms $A = D$ in $\mathcal{T}$.* □

An important service terminological representation systems provide is computing the subsumption hierarchy, i.e., computing the subconcept-superconcept relationships between the concepts of a T-box. This inferential service is usually called classification. The model-theoretic semantics introduced above allows the following formal definition of subsumption and satisfiability.

Definition 2.3 (T-box services) *Let $\mathcal{T}$ be a T-box and let C, D be concepts. Then D* subsumes *C with respect to $\mathcal{T}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$. A concept C is* satisfiable *if there is a model $\mathcal{I}$ of $\mathcal{T}$ that satisfies C, i.e., $C^{\mathcal{I}}$ is not empty.* □

[2] See [Nebel, 1989; Baader, 1990] for a treatment of cyclic definitions in concept languages.

All extensions of $\mathcal{ALCF}$ in the present paper involve attribute/role chainings, which are built from role and attribute names with the binary, associative infix operator $\circ$ which is interpreted according to

$(a, b) \in (R_1 \circ R_2)^{\mathcal{I}}$ iff
there is a c with $(a, c) \in R_1^{\mathcal{I}}$ and $(c, b) \in R_2^{\mathcal{I}}$

The special attribute name ϵ is always interpreted as identity.

In addition to the formalism defined so far, there is an assertional component (A-box) to draw terminological inferences about individuals.

Definition 2.4 (A-box syntax) *Let* OB *be an alphabet of individuals. If C is a concept and R is a role or attribute, and a, b are individuals, then*

$a = b$	(equality),
$a \neq b$	(negated equality),
$a : C$	(membership assertion), *and*
$(a, b) : R$	(role-filler assertion)

are assertional axioms. *An* A-box *is a finite set of assertional axioms.* □

An interpretation of an A-box is an interpretation $\mathcal{I}$ of $\mathcal{ALCF}$ that in addition assigns to each individual $a \in$ OB an element $a^{\mathcal{I}} \in dom(\mathcal{I})$.

An interpretation $\mathcal{I}$ *satisfies an equality (a negated equality)* $a = b$ ($a \neq b$) if $a^{\mathcal{I}} = b^{\mathcal{I}}$ ($a^{\mathcal{I}} \neq b^{\mathcal{I}}$), it *satisfies a membership assertion* $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and it *satisfies a role- or attribute-filler assertion* $(a, b) : R$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. It *satisfies an A-box $\mathcal{A}$* if it satisfies all assertional axioms in $\mathcal{A}$.

An interpretation $\mathcal{I}$ is called a *model* of $\mathcal{A}$ w.r.t. a terminology $\mathcal{T}$ if it is a model of $\mathcal{T}$ and satisfies $\mathcal{A}$.

In particular, there is no unique name assumption (UNA) ($a \neq b$ does not imply $a^{\mathcal{I}} \neq b^{\mathcal{I}}$). Since inequality assertions are allowed, the UNA can be easily simulated—if needed for selected individuals. Finally, the model-theoretic semantics is the basis for the formal specification of the reasoning services of the assertional component.

Definition 2.5 (A-box services) *Let a T-box $\mathcal{T}$ be given. An A-box is called* consistent *if it has a model w.r.t. $\mathcal{T}$. An object a is a* member *of a concept C w.r.t. an A-box $\mathcal{A}$ if all models $\mathcal{I}$ of $\mathcal{A}$ satisfy $a : C$, too.* □

Note that a is a member of C w.r.t. $\mathcal{A}$ iff $\mathcal{A} \cup \{a : \neg C\}$ is not consistent.

3 EQUALITY BASED OPERATORS

In this section a concept language based on $\mathcal{ALCF}$ with additional concept forming operators, called existential and universal role/attribute (dis)agreements,

is formally defined. These concept forming operators are based on equality and negated equality. This is quite different to the operators introduced in Section 4. As for RVMs, the subsumption problem in a concept language with these additional language constructs is undecidable.

Let $u =_{\text{RVM}} v$ be the original RVM construct, where u and v are two, possibly empty, chainings of roles and attributes. An individual a belongs to the concept $u =_{\text{RVM}} v$ iff the two sets of (transitive) role-fillers of u and v are identical. Formally, an interpretation extends to the RVMs according to:[3]

$$a \in (u =_{\text{RVM}} v)^{\mathcal{I}} \quad \text{iff} \quad a^{\mathcal{I}} u^{\mathcal{I}} = a^{\mathcal{I}} v^{\mathcal{I}}$$

Note that each of the following constructs is different from the RVM construct.

Definition 3.1 (equality-based operators) *The new concept forming operators based on equality and negated equality are defined as follows. Let u and v be two role chainings. Then*

$\forall(u = v)$ (universal agreement)
$\forall(u \neq v)$ (universal disagreement)
$\exists(u = v)$ (existential agreement)
$\exists(u \neq v)$ (existential disagreement)

are concept terms with the following semantics:

$a \in \forall(u = v)^{\mathcal{I}}$ *iff*
for all b, c with $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in v^{\mathcal{I}}$ and $(a^{\mathcal{I}}, c^{\mathcal{I}}) \in u^{\mathcal{I}}$ we have $b^{\mathcal{I}} = c^{\mathcal{I}}$

$a \in \forall(u \neq v)^{\mathcal{I}}$ *iff*
for all b, c with $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in v^{\mathcal{I}}$ and $(a^{\mathcal{I}}, c^{\mathcal{I}}) \in u^{\mathcal{I}}$ we have $b^{\mathcal{I}} \neq c^{\mathcal{I}}$

$a \in \exists(u = v)^{\mathcal{I}}$ *iff*
there exists b with $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in v^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in u^{\mathcal{I}}$

$a \in \exists(u \neq v)^{\mathcal{I}}$ *iff*
there exist b, c with $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in v^{\mathcal{I}}$ and $(a^{\mathcal{I}}, c^{\mathcal{I}}) \in u^{\mathcal{I}}$ and $b^{\mathcal{I}} \neq c^{\mathcal{I}}$ □

If u and v are attribute agreements, $u =_{\text{RVM}} v$ and $\forall(u = v)$ are equivalent concepts. This is not the case if u and/or v would contain a role. Moreover, it is at least not obvious how RVMs with roles can be simulated by the equality-based operators. Unfortunately, $\mathcal{ALCF}$ together with the constructs of the previous definition has an undecidable subsumption problem, too.

This will be shown by a reduction of the *word problem for semi-groups* to the subsumption problem in the concept language. First, the definition of the word problem is recalled. Let Σ be a finite alphabet, let Σ^* be the set of finite, possibly empty words over Σ, and let ϵ be the empty word. Then a set $S = \{l_i = r_i;\ l_i, r_i \in \Sigma^*, i = 1, \ldots, m\}$ is called a *finite presentation of a semi-group*. This set induces a binary relation $\rightarrow_S$ on Σ^*:

$u \rightarrow_S v$ iff
there are words w_1, $w_2 \in \Sigma^*$, and an $l = r \in S$ such that $u = w_1 l w_2$ and $v = w_1 r w_2$.

By $\sim_S$ we denote the reflexive, transitive, and symmetric closure of $\rightarrow_S$. It is well known that a finite presentation S exists consisting of seven equations over a two-element alphabet, $\Sigma = \{a, b\}$ say, such that it is undecidable for two words u and v whether $u \sim_S v$ holds or not (see, for instance [Boone, 1959]).

Now let this system S be given. For the two elements a, $b \in \Sigma$ two attributes a, b are introduced, respectively. Let start, left, right be additional attributes, let back, forth be additional role names, and let A be a fresh concept name. Then for two given words u, $v \in \Sigma^*$ the following concept definition schema is introduced

$$\mathsf{Eq}_{u,v} = \exists(\mathsf{left} = \mathsf{start} \circ u) \sqcap \exists(\mathsf{right} = \mathsf{start} \circ v) \sqcap \exists(\mathsf{forth} = \mathsf{start})$$

For any model $\mathcal{I}$ (of the terminology up to this point) satisfying $\mathsf{Eq}_{f_1 \cdots f_m, g_1 \cdots g_n}$ there are (not necessarily distinct) objects, $c, a_0, a_1, \cdots, a_m, b_0, b_1, \cdots, b_n$ such that

1. $(a_{i-1}, a_i) \in f_i^{\mathcal{I}}$, for $0 < i \leq m$, and $(b_{i-1}, b_i) \in g_i^{\mathcal{I}}$, for $0 < i \leq n$
2. $a_0 = b_0$, $(c, a_0) \in \mathsf{start}$, $(c, a_0) \in \mathsf{forth}$, $(c, a_m) \in \mathsf{left}$, and $(c, b_n) \in \mathsf{right}$.

This attribute/role structure is depicted in Figure 1.

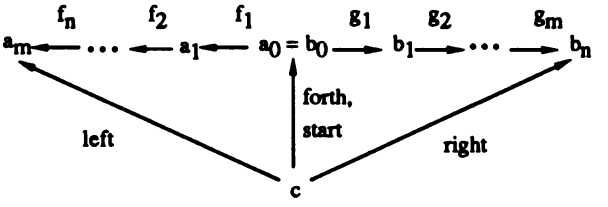


Figure 1: Representing u and v

The concept definition $\mathsf{Top} = \mathsf{A} \sqcup \neg \mathsf{A}$ introduces a name for the universal concept. A single equation $l = r \in S$ can be modeled by a concept $\mathsf{Equation}_{l=r}$ defined by the following schema:

$$\mathsf{Equation}_{l=r} = (\exists l.\mathsf{Top} \Rightarrow \exists(l = r)) \sqcap (\exists r.\mathsf{Top} \Rightarrow \exists(l = r))$$

where the expressions of the form $A \Rightarrow B$ are abbreviations for $(\neg A) \sqcup B$. The concept $\exists l.\mathsf{Top}$ is satisfied

[3] For a binary relation r and an object a the expression ar is defined as the set $\{b;\ r(a, b)\}$.

by an element $x \in dom(\mathcal{I})$ iff $l^I(x) \in dom(\mathcal{I})$, i.e., the partial funtion l^I is defined on x.

Assume that the model $\mathcal{I}$ satisfies $\mathsf{Equation}_{l=r}$ and that $a_0 \in \mathsf{Equation}^{\mathcal{I}}_{l=r}$. Then it is easy to show that $a_0 \in \mathsf{Equation}_{l=r}$, $lw \in \Sigma^*$, and $(lw)^{\mathcal{I}}(a_0) \in dom(\mathcal{I})$ implies $(rw)^{\mathcal{I}}(a_0) \in dom(\mathcal{I})$.

The presentation $S = \{e_1, \ldots, e_7\}$ can now be easily represented as

$$\mathsf{LocalS} = \mathsf{Equation}_{e_1} \sqcap \ldots \mathsf{Equation}_{e_7}.$$

But how can this restriction be imposed on each element x for which there is a $w \in \Sigma^*$ such that $w^{\mathcal{I}}(a_0) = x$? Since the concept language provides no transitive closure or cyclic definitions, the element c in Figure 1 is used as a 'relay that refreshes' the restriction. Consider, the following concept definition schema:

$$\mathsf{Loop}_\sigma = \forall \mathsf{forth}.\forall\sigma.\exists(\mathsf{back} \circ \mathsf{forth} = \epsilon) \sqcap \forall(\mathsf{forth} \circ \sigma \circ \mathsf{back} = \epsilon)$$

Any model of the concept $\mathsf{Eq}_{u,v} \sqcap \mathsf{Loop}_\mathsf{a} \sqcap \mathsf{Loop}_\mathsf{b}$ leads

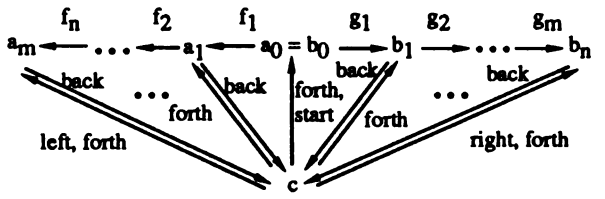


Figure 2: Repeating back and forth

to a role/attribute structure as depicted in Figure 2. More precisely, c has all the x as role fillers for forth that can be reached from a_0 by a word $w \in \Sigma^*$. Now it is easy to impose the requirements of S on each of these elements: $\mathsf{GlobalS} = \forall\mathsf{forth}.\mathsf{LocalS}$

Proposition 3.2 *Given two words $u, v \in \Sigma^*$*

$\mathsf{Eq}_{u,v} \sqcap \mathsf{Loop}_\mathsf{a} \sqcap \mathsf{Loop}_\mathsf{b} \sqcap \mathsf{GlobalS}$ *subsumes* $\exists(\mathsf{left} = \mathsf{right})$

iff $u \sim_S v$.

Proof.
1) *Assume that $u \sim_S v$:*
Let $\mathcal{I}$ be a model of the above concept definitions, and let c be in $(\mathsf{Eq}_{u,v} \sqcap \mathsf{Loop}_\mathsf{a} \sqcap \mathsf{Loop}_\mathsf{b} \sqcap \mathsf{GlobalS})^{\mathcal{I}}$. Relying on the above construction it is easy to prove the following:

If $w^{\mathcal{I}}$ is defined on $\mathsf{start}^{\mathcal{I}}(\mathsf{c})$, and $w \rightarrow_S w'$ or $w' \rightarrow_S w$ then $w^{\mathcal{I}}(\mathsf{start}^{\mathcal{I}}(\mathsf{c})) = w'^{\mathcal{I}}(\mathsf{start}^{\mathcal{I}}(\mathsf{c}))$.

By definition there is a finite derivation of $u \sim_S v$ in terms of the symmetric closure of $\rightarrow_S$ and thus, $u^{\mathcal{I}}(\mathsf{start}^{\mathcal{I}}(\mathsf{c})) = v^{\mathcal{I}}(\mathsf{start}^{\mathcal{I}}(\mathsf{c}))$ and $\mathsf{left}^{\mathcal{I}}(c) = \mathsf{right}^{\mathcal{I}}(c)$. By definition: $c \in \exists(\mathsf{left} = \mathsf{right})$.

2) *Assume that not $u \sim_S v$:*
It is easy to verify that the interpretation constructed below is a model of the above concept definitions and a counter example to the subsumption relation in question.

Let $dom(\mathcal{I}) = \Sigma^*/_{\sim_S} \cup \{c\}$ where $\Sigma^*/_{\sim_S}$ is the set of equivalence classes induced by the congruence relation $\sim_S$. The partial functions $\mathsf{a}^{\mathcal{I}}$ and $\mathsf{b}^{\mathcal{I}}$ are defined as left multiplications for all $[x] \in \Sigma^*/_{\sim_S}$:

$$\mathsf{a}^{\mathcal{I}}([x]) = [ax] \text{ and } \mathsf{b}^{\mathcal{I}}([x]) = [bx].$$

The other roles and attributes are defined as suggested by the construction:

1. $\mathsf{left}^{\mathcal{I}}(c) = [u]$, $\mathsf{right}^{\mathcal{I}}(c) = [v]$, and $\mathsf{start}^{\mathcal{I}}(c) = [\epsilon]$,
2. $(c, x) \in \mathsf{forth}^{\mathcal{I}}$, for every $x \in \Sigma^*/_{\sim_S}$, and
3. $(x, y) \in \mathsf{back}^{\mathcal{I}}$ if $(y, x) \in \mathsf{forth}^{\mathcal{I}}$. □

Corollary 3.3 *The subsumption problem in a concept language based on $\mathcal{ALCF}$ and extended by the* equality-based operators *universal and existential agreement is undecidable.* □

This result shows that, as long as equality is involved, it is wise to restrict oneselfs to attribute (dis)agreements.

4 OPERATORS WITH PREDICATES

It is easy to see (for example by a comparison with CLASSIC) that the subsumption problem remains decidable if the equality-based operators are restricted to chainings of attributes. The reduction in the undecidability proof in the previous section relied heavily on the possibility to specify cyclic role structures (for instance, $\exists(\mathsf{back} \circ \mathsf{forth} = \epsilon)$).

In this section two ideas are developed that remove the capability to specify this kind of cyclic structure from the concept language. The first idea is to replace the equality in the equality-based operators by uninterpreted, possibly negated n-ary predicate symbols. The second idea is to split the interpretation domain into two separate domains: the *abstract* and the *concrete domain* [Baader and Hanschke, 1991a]. Role and attribute fillers can now be restricted by predicates of the concrete domain, too. But concepts are always subsets of the concrete domain.

Together with attribute (dis)agreements the abstract and the concrete predicate based operators are a powerful, still decidable, means to specify structural properties.

4.1 CONCRETE DOMAINS

Before the concept forming operators are introduced the notion "concrete domain" has to be formalized.

Definition 4.1 *A* concrete domain $\mathcal{D}$ *consists of a set* $dom(\mathcal{D})$*, the domain of* $\mathcal{D}$*, and a set* $pred(\mathcal{D})$*, the predicate names of* $\mathcal{D}$*. Each predicate name* p *is associated with an arity* n*, and an* n*-ary predicate* $p^{\mathcal{D}} \subseteq dom(\mathcal{D})^n$. □

An important example is the concrete domain $\mathcal{R}$ of real arithmetic. The domain of $\mathcal{R}$ is the set of all real numbers, and the predicates of $\mathcal{R}$ are given by formulae which are built by first order means (i.e., by using logical connectives and quantifiers) from equalities and inequalities between integer polynomials in several indeterminates.[4] For example, $x + z^2 = y$ is an equality between the polynomials $p(x, z) = x + z^2$ and $q(y) = y$; and $x > y$ is an inequality between very simple polynomials. From these equalities and inequalities one can e.g. build the formulae $\exists z(x + z^2 = y)$ and $\exists z(x + z^2 = y) \vee (x > y)$. The first formula yields a predicate name of arity 2 (since it has two free variables), and it is easy to see that the associated predicate is $\{(r, s)$; r and s are real numbers and $r \leq s\}$. Consequently, the predicate associated to the second formula is $\{(r, s)$; r and s are real numbers$\}$ $= dom(\mathcal{R}) \times dom(\mathcal{R})$.

To get inference algorithms for the extended concept language which will be introduced below, the concrete domain has to satisfy some additional properties.

For technical reasons the set of predicate names of the concrete domain is required to be *closed under negation*, e.g., if p is an n-ary predicate name in $pred(\mathcal{D})$ then a predicate name q in $pred(\mathcal{D})$ has to exist such that $q^{\mathcal{D}} = dom(\mathcal{D})^n \setminus p^{\mathcal{D}}$. In addition, a unary predicate name is needed which denotes the predicate $dom(\mathcal{D})$.

The property which will be formulated now clarifies what kind of reasoning mechanisms are required in the concrete domain. Let p_1, ..., p_k be k (not necessarily different) predicate names in $pred(\mathcal{D})$ of arities n_1, ..., n_k. Consider the conjunction

$$\bigwedge_{i=1}^{k} p_i(\underline{x}^{(i)}).$$

Here $\underline{x}^{(i)}$ stands for an n_i-tuple $(x_1^{(i)}, ..., x_{n_i}^{(i)})$ of variables. It is important to note that neither all variables in one tuple nor those in different tuples are assumed to be distinct. Such a conjunction is said to be *satisfiable* iff there exists an assignment of elements of $dom(\mathcal{D})$ to the variables such that the conjunction becomes true in $\mathcal{D}$.

For example, let $p_1(x, y)$ be the predicate $\exists z(x + z^2 = y)$ in $pred(\mathcal{R})$, and let $p_2(x, y)$ be the predicate $x > y$ in $pred(\mathcal{R})$. Obviously, neither the conjunction $p_1(x, y) \wedge p_2(x, y)$ nor $p_2(x, x)$ is satisfiable.

Definition 4.2 *A concrete domain* $\mathcal{D}$ *is called* admissible *iff* (i) *the set of its predicate names is closed under negation and contains a name for* $dom(\mathcal{D})$*, and* (ii) *the satisfiability problem for finite conjunctions of the above mentioned form is decidable.* □

The concrete domain $\mathcal{R}$ is admissible. This is a consequence of Tarski's decidability result for real arithmetic [Tarski, 1951; Collins, 1975]. For the linear case (where the polynomials in the equalities and inequalities have to be linear) there exist more efficient methods (see e.g. [Weispfenning, 1988; Loos and Weispfenning, 1990]).

4.2 THE ADDITIONAL OPERATORS

With the above formalization of concrete domains the extension $\mathcal{ALCFP}(\mathcal{D})$ of $\mathcal{ALCF}$ which is parametrized by an admissible concrete domain $\mathcal{D}$ can be defined. The new concept forming operators can be seen as generalizations of the value restriction and the exists-in restriction.

Definition 4.3 (syntax of $\mathcal{ALCFP}(\mathcal{D})$) *The concept formalism of* $\mathcal{ALCF}$ *is extended by the following new concept forming operators. Let* $u_1, \cdots, u_n$ *be role/attribute chainings. Then*

$\forall u_1, \cdots, u_n.\rho$ (generalized value restriction)
$\exists u_1, \cdots, u_n.\rho$ (generalized exists-in restriction)

are concept terms in each of the following cases: The term ρ *which is called* restrictor

1. *is a predicate of the concrete domain with arity* n,
2. *is of the form* p *or* $\neg p$*, where* p *is an abstract predicate of arity* n,
3. *is a concept term and* $n = 1$*, or*
4. *is "*$=$*" or "*$\neq$*",* n *is* 2*, and* u_1, u_2 *are chainings of* **attributes**. □

In addition to defining the interpretation of the new operators, the interpretation function has to take care of the concrete domain. This somehow makes the definition complicated at first glance.

Definition 4.4 (semantics of $\mathcal{ALCFP}(\mathcal{D})$) *The differences of interpretations of* $\mathcal{ALCF}$ *and the extended language are as follows:*

The set $dom(\mathcal{I})$*, which is called* abstract domain *for this language, is required to be disjoint to* $dom(\mathcal{D})$.

[4] For the sake of simplicity it is assumed here that the formula itself is the predicate name. In applications, the user will probably take his own intuitive names for these predicates.

Because attributes and roles link the abstract with the concrete domain their interpretation is liberated: An attribute f is interpreted as a partial function

$$f^{\mathcal{I}} : dom(\mathcal{I}) \longrightarrow dom(\mathcal{I}) \cup dom(\mathcal{D})$$

and a role r as a binary predicate

$$r^{\mathcal{I}} \subseteq dom(\mathcal{I}) \times (dom(\mathcal{I}) \cup dom(\mathcal{D})).$$

An abstract predicate p of arity n is interpreted as $p^{\mathcal{I}} \subseteq dom(\mathcal{I})^n$ and $(\neg p)^{\mathcal{I}}$ as $dom(\mathcal{I})^n \setminus p^{\mathcal{I}}$, and a concrete predicate p is interpreted as $p^{\mathcal{I}} = p^{\mathcal{D}}$.

It remains to define how the new operators are interpreted:

$a \in (\forall u_1, \cdots, u_n.\rho)^{\mathcal{I}}$ *iff*
for all $y_1, \cdots, y_n$ *with* $(x, y_1) \in u_1^{\mathcal{I}}, \cdots,$ $(x, y_n) \in u_n^{\mathcal{I}}$ *we have* $(y_1, \cdots, y_n) \in \rho^{\mathcal{I}}$

$a \in (\exists u_1, \cdots, u_n.\rho)^{\mathcal{I}}$ *iff*
there exists $y_1, \cdots, y_n$ *with* $(x, y_1) \in u_1^{\mathcal{I}}$, $\cdots$, $(x, y_n) \in u_n^{\mathcal{I}}$ *and* $(y_1, \cdots, y_n) \in \rho^{\mathcal{I}}$

□

The assertional component of the extended formalism allows additional forms of assertional axioms. These are the *predicate assertions* $\rho(a_1, \cdots, a_n)$ where ρ can be an abstract, a negated abstract, or a concrete predicate of arity n. They are satisfied by an interpretation $\mathcal{I}$ iff $(a_1^{\mathcal{I}}, \cdots, n^{\mathcal{I}}) \in \rho^{\mathcal{I}}$.

As already mentioned the membership problem can be reduced to a consistency test. Since a concept term C subsumes a concept term D iff $a : \neg C \sqcap D$ is inconsistent, the subsumption problem can also be reduced to a consistency test. The next section presents a decision procedure that answers in finite time the consistency problem of this extended language. So, the following theorem holds:

Theorem 4.5 *Assume that an admissible concrete domain $\mathcal{D}$ is given. Then there exist decision procedures for the consistency, the subsumption, and the membership problem in $\mathcal{ALCFP}(\mathcal{D})$.* □

5 THE BASIC ALGORITHM

This section presents an algorithm that decides in finite time whether a given A-box $\mathcal{A}_0$ is consistent or not. The algorithm is a generalization of the technique that was introduced in [Schmidt-Schauß and Smolka, 1991] and further elaborated, e.g., in [Baader and Hanschke, 1991b; Baader, 1991; Hollunder *et al.*, 1990]

Roughly, the algorithm proceeds as follows. It starts with a given A-box $\mathcal{A}$, and applies transformation rules to $\mathcal{A}$ that make the knowledge represented by the assertions more explicit. Ultimately, one of the following two situations occurs:

1. The A-box becomes "obviously contradictory", or
2. all knowledge has been made explicit.

In the latter case the A-box is called complete and describes directly a model of the original $\mathcal{A}$. In the other case $\mathcal{A}$ is inconsistent.

Sometimes it is necessary to make a case distinction during the transformation process, since disjunctions occur (implicitly and explicitly) in the formalism. So, a transformation step may transform a single A-box $\mathcal{A}$ in two new A-boxes $\mathcal{B}_1$ and $\mathcal{B}_2$. In this case $\mathcal{A}$ is inconsistent if both $\mathcal{B}_1$ and $\mathcal{B}_2$ are inconsistent. For that reason, the algorithm operates with sets of A-boxes rather than a single A-box. If the consistency of an A-box $\mathcal{A}$ has to be checked, the algorithm is initialized with the singleton set $\mathcal{M}_0 = \{\mathcal{A}_0\}$ where $\mathcal{A}_0$ is the unfolded (see below) version of $\mathcal{A}$.

5.1 UNFOLDING

Let a terminology $\mathcal{T}$ and an A-box $\mathcal{A}_0$ be given. To simplify the presentation, the A-boxes are first normalized by the *unfolding rule*. It replaces a concept name C by its definition t if the concept definition $C = t$ is in $\mathcal{T}$. Because terminologies do not contain cycles this rule can only be applied finitely many times. After all the defined concepts have been replaced, the terminology is not needed any more for the consistency test.

5.2 TRANSFORMATION RULES

This section presents the transformation rules that operate on the set $\mathcal{M}_0$. They generate a finite sequence (see the next section for a proof of the finiteness) of sets $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, ... $\mathcal{M}_k$ of A-boxes.

5.2.1 Pushing Negation

The *negation rules* propagate negation ("¬") towards the leaves of the concept terms. Recall that ¬ is a complement operator w.r.t. $dom(\mathcal{I})$ and that attributes and roles link the abstract domain with the concrete domain. So it is convenient to introduce a global complement operator $\sim$. It is defined by $\sim\rho^{\mathcal{I}} = (dom(\mathcal{I}) \cup dom(\mathcal{D})^n \setminus \rho^{\mathcal{I}}$ where ρ is a restrictor with arity n (Definition 4.3).

$$\frac{\neg\neg s}{s}, \quad \frac{\neg(s \sqcap t)}{\neg s \sqcup \neg t}, \quad \frac{\neg(s \sqcup t)}{\neg s \sqcap \neg t}, \quad \frac{\sim\sim\rho}{\rho}$$

$$\frac{\neg\forall v_1, \cdots, v_n.\rho}{\exists v_1 \cdots v_n.\sim\rho}, \quad \frac{\neg\exists v_1, \cdots, v_n.\rho}{\forall v_1 \cdots v_n.\sim\rho}$$

5.2.2 Transformation Rules

Whereas the previous rules are rewriting rules that operate on (sub)terms the following rules operate on

the level of assertions. For these remaining rules the expressions of the form

$$\frac{\textit{premises}}{\textit{consequences}}$$

have to be read as follows: if there is an A-box $\mathcal{A}$ in the current $\mathcal{M}_i$ that fulfills the premises, then the successor $\mathcal{M}_{i+1}$ is obtained by adding the consequences to $\mathcal{A}$. A rule must not be applied in an A-box $\mathcal{A}$ with a particular instantiation of the premises if the rule has already been applied with the same instantiation of the premises in that A-box. Neither must it be applied if the A-box contains an obvious contradiction (see Section (5.2.4).

If vertical bars "|" occur in the consequence of a rule, this means that the A-box $\mathcal{A} \in \mathcal{M}_i$ to which the rule is applied has to be replaced with new A-boxes for each of the alternatives that are separated by the bar(s). So, in these cases $\mathcal{M}_{i+1}$ contains more A-boxes than its predecessor $\mathcal{M}_i$.

The Operator Rules

These rules split concept terms into its immediate subterms and generate new assertions.

$$(\mathbf{R}\sqcap)\ \frac{a : s \sqcap t}{a : s, a : t}$$

$$(\mathbf{R}\sqcup)\ \frac{s \sqcup a : t}{a : s \mid a : t}$$

This rule transforms the affected A-box into two A-boxes.

$$(\mathbf{R}\forall)\ \frac{(a, b_1) : v_1, \cdots, (a, b_n) : v_n, a : \forall v_1 \cdots v_n.\rho}{(b_1, \cdots, b_n) : \rho}$$

A premise $(a, b) : v$ is fulfilled if

1. $a = b$ and v is ϵ or
2. there is a $(a, c) : R$ in the A-box, v splits into Rv' where R is an attribute or role, and, recursively, $(c, d) : v'$ is fulfilled.

$$(\mathbf{R}\exists)\ \frac{a : \exists v_1 \cdots v_n.\rho}{(a, b_1) : v_1, \cdots, (a, b_n) : v_n, (b_1, \cdots, b_n) : \rho}$$

Here the b_i are fresh individuals.

The Role and Attribute Rules

The (R$\exists$) rule may generate new assertions of the form $(a, b) : v$ where v is a chaining of attributes or roles. It may also cause *forks*. These are pairs of attribute-filler assertions $(a, b) : f$, $(a, c) : f$ with $b \neq c$. These configurations are treated by the following rules:

$$(\mathbf{R}\circ)\ \frac{(a, b) : R \circ v}{(a, c) : R, (c, b) : v}$$

Here c is a fresh individual.

$$(\mathbf{R}\epsilon)\ \frac{(a, b) : \epsilon}{a = b}$$

$$(\mathbf{R}\rightarrowtail)\ \frac{(a, b_1) : f, (a, b_2) : f}{b_1 = b_2}$$ if f is an attribute.

The $\sim$ Rules

The following rules deal with the global complement operator if it occurs at the top level in an assertion.

$$(\mathbf{R}\sim\mathcal{D})\ \frac{(a_1, \cdots, a_n) : \sim\rho}{a_1 : \mathsf{T} \mid \cdots \mid a_n : \mathsf{T} \mid (a_1, \cdots, a_n) : \overline{\rho}}$$

if ρ is a concrete predicate and $\overline{\rho}$ is the complement of ρ w.r.t. $dom(\mathcal{D})$ (since $\mathcal{D}$ is admissible $\overline{\rho}$ is also a predicate of the concrete domain), and T is a specific concept name that is always interpreted as $dom(I)$.

$$(\mathbf{R}\sim\mathcal{P})\ \frac{(a_1, \cdots, a_n) : \sim\rho}{a_1 : \mathcal{D} \mid \cdots \mid a_n : \mathcal{D} \mid (a_1, \cdots, a_n) : \neg\rho}$$

if ρ is a concept term and $n = 1$ or its is an abstract predicate.

$$(\mathbf{R}\sim=)\ \frac{(a_1, a_2) : \sim=}{(a_1, a_2) : \neq} \qquad (\mathbf{R}\sim\neq)\ \frac{(a_1, a_2) : \sim\neq}{(a_1, a_2) : =}$$

The Domain Rules

The abstract and the concrete domain are disjoint. This may lead to obvious contradictions. The domain rules try to make explicit the domain to which an individual belongs.

$$(\mathbf{R}\mathcal{P}\mathsf{T})\ \frac{(a_1, \cdots, a_n) : \rho}{a_1 : \mathsf{T}, \cdots, a_n : \mathsf{T}}$$

if ρ is a primitive concept or an abstract predicate.

$$(\mathbf{R}R\mathsf{T})\ \frac{(a_1, a_2) : R}{a_1 : \mathsf{T}}$$ if R is a role or attribute.

$$(\mathbf{R}\mathcal{D}\mathsf{T})\ \frac{(a_1, \cdots, a_n) : \rho}{a_1 : \mathcal{D}, \cdots, a_n : \mathcal{D}}$$

if ρ is a concrete predicate different from $\mathcal{D}$.

The Identification Rule

The attribute agreements and the functional character of the attributes may lead to equality assertions. These are treated by the following rule:

$$(\mathbf{R}=)\ \frac{(a, b) : =}{\text{replace } a \text{ by } b \text{ in the affected A-box}}$$

5.2.3 The Strategy

In order to get a terminating algorithm the order in which the rules may be executed has to be restricted. Identifications of individuals have to take place as soon as possible. So the "role and attribute rules" and the identification rule are executed with the highest priority.

If a transformation rule has been applied to some assertions and later some individuals in these assertions

are replaced during applications of the (R=) rule, the transformation rule must not be applied again to these assertions (although the premises are not exactly the same).

5.2.4 Obvious Contradictions

A single A-box $\mathcal{A}$ is *obviously contradictory* in each of the following cases:

Primitive Clash: The A-box contains a pair of assertions of the form $\underline{a} : \rho, \underline{a} : \neg\rho$ where ρ is an abstract predicate (resp., a concept term) and $\underline{a}$ is a tuple of individuals (resp., an individual).

Domain Clash: The A-box contains $a : \top, a : \mathcal{D}$.

Equality Clash: The A-box contains $a \neq a$.

Concrete Domain Clash: The A-box contains predicate assertions $\underline{a}_1 : p_1, \ldots, \underline{a}_n : p_n$ where the p_i are concrete predicates and the satisfiability test of the concrete domain says that the above conjunction is not satisfiable.

5.3 SUMMARY OF ALGORITHM

The following procedure, written in a pseudo programming language, summarizes the consistency test of A-boxes of $\mathcal{ALCFP}(\mathcal{D})$.

Algorithm 5.1 (consistency test) *The procedure takes an A-box $\mathcal{A}$ as an argument and checks whether it is consistent or not.*

```
define procedure check-consistency(A)
  A_0 := unfold(A)
  r := 0
  M_0 := {A_0}
  while 'a transformation rule is applicable to M_r'
  do
    r := r + 1
    M_r := apply-a-transformation-rule(M_{r-1})
  od
  if 'there is an A ∈ M_r that is not obviously
    contradictory'
  then return consistent
  else return inconsistent
```

□

6 THE PROOF

In this section termination, soundness, and completeness of the consistency test (Algorithm 5.1) are proved. Together, these facts imply that the algorithm is a decision procedure for the consistency of an A-box $\mathcal{A}$.

Proposition 6.1 *Assume that Algorithm 5.1 is applied to $\mathcal{A}$. Then*

1. *the algorithm always computes in finite time a set $\mathcal{M}_r$ of A-boxes each of which is complete or obviously contradictory, and*
2. *the initial A-box is inconsistent iff all A-boxes $\mathcal{A} \in \mathcal{M}_r$ contain a clash.*

Proof. The proposition is a consequence of the four lemmata (6.2, 6.3, 6.4) stated and proved below. □

As already mentioned, unfolding terminates, because terminologies are acyclic. Since unfolding does not change the satisfiability of an A-box, this preparatory step is neglected in the proof.

The while loop of Algorithm 5.1 reduces the semantic problem of consistency for the A-box $\mathcal{A}_0$ to a simple syntactic problem for a finite set $\mathcal{M}_r$ of A-boxes. This syntactic problem is to check whether there is an A-box in $\mathcal{M}_r$ that is not obviously contradictory. In order to show the correctness of the reduction, termination is proved first.

Assume that a computation using the algorithm is given and that in a single execution of the loop body the A-boxes $\mathcal{B}_1, \ldots, \mathcal{B}_n$, $n > 0$, have been derived by an application of one of the transformation rules to an A-box $\mathcal{A}$. Then the $\mathcal{B}_i$ are called *descendants* of $\mathcal{A}$.

Lemma 6.2 (termination) *The algorithm always computes a complete set of A-boxes $\mathcal{M}_r$ in finite time.*

Proof. Assume that a possibly infinite computation is given. In order to show termination it suffices to prove that there is no infinite sequence of A-boxes $\mathcal{A}_0, \mathcal{A}_1, \ldots$ where $\mathcal{A}_{i+1}$ is a descendant of $\mathcal{A}_i$.

This sequence with the associated applications of transformation rules defines a sequence of trees δ_0, δ_1, ... as follows:

1. The initial tree δ_0 consists just of the edges $\beta \rightarrow \nu(a : C)$ where $a : C$ is a membership assertion in $\mathcal{A}_0$ and β is an additional root.
2. For an individual a let $a^{(k)}$ denote the individual name that stands in place of a after all replacements of the rule (R=) up to $\mathcal{A}_k$ have been performed.

   Each time a transformation rule is applied to an A-box, $\mathcal{A}_k$ say, generating new membership assertions $b_j : B_j$, $j = 1, \ldots, l$, the tree δ_{i+1} is constructed from δ_i. This is done by adding edges $\nu(a : A) \rightarrow \nu(b_j : B_j)$, $j = 1, \ldots, l$, where the $\nu(b_j : B_j)$ are new nodes and $a^{(k)} : A$ occurs in the (instantiated) premise of the transformation rule (there is always exactly one such assertion).

If individuals are replaced, this is not done in the tree. So δ_i conservatively extends δ_{i+1}.

Note that not every application of a transformation rule leads to a new δ_i (consider for example the rule

(R↦). But, it is easy to observe that the computation of an infinite sequence of descending A-boxes (as the one above) leads to an infinite sequence of trees with an increasing number of nodes.

If the δ's are considered as sets of edges,

$$\Delta = \bigcup_{i=0,1,2,\ldots} \delta_i$$

is a tree, too. If it can be shown that Δ is finite, this yields a contradiction and the proof is done.

Assume that Δ is infinite.

1) The mapping $|.|$ from nodes to naturals is inductively defined as

1. $|\nu(b : B)| := |B|$
2. $|\rho| := 1$, if ρ is an abstract or a concrete predicate, a primitive concept, $=$, or $\neq$.
3. $|B \sqcap C| := |A| + |B|$,
   $|B \sqcup C| := |A| + |B|$,
   $|\exists v_1, \ldots, v_n.\rho| := |\rho| + 1$,
   $|\forall v_1, \ldots, v_n.\rho| := |\rho| + 1$,
   $|\sim\rho| := |\rho| * 2 + 1$,
   $|\neg\rho| := |\rho| * 2$

has the following nice property: $\nu(a : A) \rightarrow \nu(b : B)$ implies $|\nu(a : A)| > |\nu(b : B)|$.

This implies that the depth of Δ (defined as the number of edges in the longest directed path) is bounded by

$$\max\{|\nu(a : C)|;\ \nu(a : C) \text{ occurs in } \delta_0\} + 1.$$

2) It remains to show that the tree is finitely branching. Let a node $\nu(a : C)$ be given. If C is not a generalized value restriction, the node has exactly one descendant. This holds because the transformation rule applied to the assertion related to this node has exactly on assertion in its premise, and, by definition, rules are only applied once per premise.

If C is a generalized-value restriction $\forall u_1, \ldots, u_n.\rho$, where ρ is not a concept term then the node does not have any successor, because only new membership assertions lead to new nodes.

So a node $\nu(a : C)$ where C is a value restriction $\forall R.C$ is the last kind of node that could have infinitely many immediate successors. The rule (R$\forall$) is applied once to this assertion per attribute-filler or role-filler assertion $(a, b) : R$.

Since the "role and attribute rules" are executed with a priority higher than the priority of (R$\forall$), the node has only one descendant, too, if R is an *attribute*.

Let a node ν of the form $\nu(a_0 : \forall R.C)$ be given where R is a *role* and C is a concept. Note that this is the last remaining case. All other kinds of nodes have already been proved to have only finitly many successors. Assume that ν has infinitely man descendants.

Observation 1: To get these infinitely many descendants the computation has to generate infinitely many role-filler assertions of the form $(a_0, b) : R$. These come

1. either from an assertion $c : \exists u_1, \ldots, u_n.\rho$ or
2. an assertion $(a', b) : R$ has been generated from an assertion $c : \exists u_1, \ldots, u_n.\rho$ and, later, the individuals $'a$ and a_0 have been identified.

In both cases c is linked to a_0 through a directed path labeled with attributes. Let N be the infinite set of nodes belonging to the generating assertions $c : \exists u_1, \ldots, u_n.\rho$, and let F be the infinite set of generated R role fillers of a_0.

Note that there can be only finitely many exceptions $(a_0, b) : R$ that are not generated from a node in N.

Observation 2: Let $b \in F$. Now observe that all individuals d that can be reached from b through a directed path of attribute/role assertions cannot be reached from another $b' \in F$, $b' \neq b$.

Observation 3: If $\nu(a : A)$ is any node in Δ and $\nu(b : B)$ is any other node below $\nu(a : A)$ then a equals b or there is a directed path from a to b.

Let $(a_0, b) : R$, $b \in F$ be one of the generated assertions. Then there does not exist an attribute/role path from b to a_0.

Together with the contraposition of Observation 3 this implies that the infinitely many nodes in N are not descendants of ν in Δ.

These infinitely many nodes must be descendants of the finitely many nodes in $\mathcal{A}_0$. Consider the subtree Δ' of Δ that is obtained by taking all pathes from the nodes in D to the root β.

Since Δ has finite depth, Δ' has finite depth. Now assume that there is an infinite branch in Δ' at a node $\nu(b : B)$. Then, as above, B is a value restriction $\forall R.'C'$ with a role R', and there are infinitely many role-fillers for b. Only finitely many are not generated by the rules (R$\exists$) and (R$\circ$).

So, there are at least two nodes in the infinite N that are in different subtrees of $\nu(b : B)$ belonging to *generated* role-fillers b_1 and b_n of b w.r.t. R'.

Analog to Observation 2, the set of individuals reachable from b_1 and b_2, respectively, are disjoint. But a_0 is reachable from both b_1 and b_2 because of observations 3 and 1: contradiction. So, Δ' is finite which contradicts the infinity of N. So, ν has only finitely many descendants and Δ is finite,too. □

To prove the second part of Proposition 6.1, the notion of *contradictory A-boxes* is introduced. It is the syntactic equivalent to inconsistent A-boxes. The definition

is by induction on the relation "descendant" which has just been proved noetherian. An A-box $\mathcal{A}$ occurring in the computation is *contradictory* with respect to a computation iff

- $\mathcal{A}$ does not have descendants and is obviously contradictory, or
- all descendants of $\mathcal{A}$ are contradictory.

Please note that according to this definition $\mathcal{A}_0$ is contradictory iff after the loop in Algorithm 5.1 all A-boxes in $\mathcal{M}_r$ are obviously contradictory.

Lemma 6.3 (soundness) *An A-box that is contradictory with respect to a given computation is inconsistent.*

Proof. The proof is by induction on the definition of *contradictory*, with a case analysis according to the transformation rule applied. Assume that a contradictory A-box $\mathcal{A}$ is given. It has to be shown that it does not have a model.

1) If $\mathcal{A}$ does not have a descendant, it must be obviously contradictory and cannot have a model.

2) For the induction step, assume to the contrary that $\mathcal{A}$ has a model $\mathcal{I}$. It has to be shown that at least one of the descendants of $\mathcal{A}$ has a model. This will be a contradiction to the induction hypothesis, because all descendants of contradictory A-boxes are contradictory.

This shall only be demonstrated for the case of the (R$\forall$) rule. The other cases can be treated similarly.

Assume that the rule has been applied to the axioms $(a,b_1):v_1, \cdots, (a,b_n):v_n, a:\forall v_1 \cdots v_n.\rho$ generating the descendant $\mathcal{B}$. Please note that $\mathcal{B}$ is a superset of $\mathcal{A}$ and that the only axiom in $\mathcal{B}$ that is not in $\mathcal{A}$ is $(b_1,\cdots,b_n):\rho$. Hence, it suffices to show that $\mathcal{I}$ satisfies $b:C$. This is an immediate consequence of the definition of the generalized value restriction. □

Lemma 6.4 (completeness) *If the initial A-box A_0 is not contradictory with respect to a given computation, it has a model.*

Proof. If $\mathcal{A}_0$ is not contradictory then there is an A-box $\mathcal{B} \supseteq \mathcal{A}_0$ in $\mathcal{M}_r$ that is not obviously contradictory. Next an interpretation $\mathcal{I}$ of $\mathcal{B}$ is defined:

1. Because the clash rule related to the concrete domain is not applicable, there is a variable assignment α that satisfies the conjunction of all occurring axioms of the form $P(x_1,\ldots,x_n)$. The interpretation $\mathcal{I}$ interprets all x with $x:\mathcal{D}$ in $\mathcal{B}$ as $\alpha(x)$.
2. The domain $dom(\mathcal{I})$ consists of all the objects x with $x:\top$ in $\mathcal{B}$.
3. Let ρ be a primitive concept or an abstract predicate. Then $(a_1,\ldots,a_n) \in \rho^{\mathcal{I}}$ iff $(a_1,\ldots,a_n):\rho$ occurs in $\mathcal{B}$. The domain rules ensure that all a_i belong to $dom(\mathcal{I})$.
4. Let R be a role or attribute. Then $(a,b) \in R^I$ iff $(a,b):R$ is in $\mathcal{B}$. This is well defined even if R is an attribute, because of the transformation rule (R$\mapsto$), which is not applicable to $\mathcal{B}$. The domain rules ensure that a belongs to $dom(\mathcal{I})$.

It is straightforward, but tedious, to show by induction on the size of the axioms that $\mathcal{I}$ is not only an interpretation but also a model of $\mathcal{B}$.

Here only the case of the generalized value restriction is demonstrated:

> Assume $a:\forall v_1 \cdots v_n.\rho$ is in $\mathcal{B}$. Let any objects $b_1, \ldots, b_n$ be given. If $(a,b_1) \in v_1^{\mathcal{I}}$, $\cdots$, $(a,b_n) \in v_n^{\mathcal{I}}$ the transformation rule (R$\forall$) ensures that $(b_1,\ldots,b_n):\rho$ is in $\mathcal{B}$. By induction hypothesis, $\mathcal{I}$ satisfies this assertion. Since the b_i were arbitrary, by definition, $\mathcal{I}$ satisfies the generalized value restriction.

Finally, $\mathcal{A}_0 \subseteq \mathcal{B}$ is used to deduce that $\mathcal{I}$ is also a model for $\mathcal{A}_0$ □

7 CONCLUSIONS

In [Schmolze, 1989] a family of concept languages is presented which is also based on n-ary predicates. One motivation for this formalism is that some concepts are more naturally expressed in terms of n-ary predicates. The terminological formalisms of the present paper are more "object centered" and use predicates only to specify role interaction. It is not clear for which members of the family of concept languages presented in [Schmolze, 1989] decision procedures for the common reasoning services exist.

In the present paper concept forming operators to specify interaction of roles and attributes have been studied. It has been shown that universal and existential role/attribute (dis)agreements lead in general to an undecidable subsumption problem. Nevertheless, an expressive concept language with sound and complete reasoning algorithms has been presented that allows to specify interactions on the basis of abstract and concrete predicates as well as attribute (dis)agreements.

The TAXON system implements a superset of the concept language $\mathcal{ALCFP}(\mathcal{D})$. The system is written in CommonLisp and has been implemented in the ARC-TEC project (Acquisition and Representation and Compilation of TEChnical Knowledge) [Bernardi *et al.*, 1991]. It is mainly used in an application in mechanical engineering (ARC-TEC) dealing with the production planning of lathe workpieces and a project

TOOCON (TOOls for model-based CONfiguration) that develops a configuration system for low-voltage switch boards.

References

[Baader and Hanschke, 1991a]
F. Baader and Ph. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of the* 12^{th} *International Joint Conference on Artificial Intelligence*, 1991.

[Baader and Hanschke, 1991b]
F. Baader and Ph. Hanschke. A scheme for integrating concrete domains into concept languages. Research Report RR-91-10, DFKI, 1991.

[Baader, 1990]
F. Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, volume 2, pages 621–626, 1990.

[Baader, 1991]
F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proceedings of the* 12^{th} *International Joint Conference on Artificial Intelligence*, 1991.

[Bernardi *et al.*, 1991]
A. Bernardi, H. Boley, K. Hinkelmann, Ph. Hanschke andh C. Klauck, O. Kühn, R. Legleitner, M. Meyer, M.M. Richter, G. Schmidt, F. Schmalhofer, and W. Sommer. ARC-TEC: Acquisition, Representation and Compilation of Technical Knowledge. In *Expert Systems and their Applications: Tools, Techniques and Methods*, 1991.

[Boone, 1959]
W. W. Boone. The word problem. *Ann. of Mat.*, 1959.

[Borgida *et al.*, 1989]
A. Borgida, R. Brachman, D. McGuinness, and L. Resnick. CLASSIC: A structural data model for objects. In *International Conference on Managment on Data*. ACM SIGMOD, 1989.

[Brachman and Schmolze, 1985]
R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2), 1985.

[Collins, 1975]
G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *2nd Conference on Automata Theory & Formal Languages*, volume 33, 1975.

[Hollunder *et al.*, 1990]
B. Hollunder, W. Nutt, and M. Schmidt-Schauß. Subsumption algorithms for concept description languages. In *9th European Conference on Artificial Intelligence (ECAI'90)*, 1990.

[Loos and Weispfenning, 1990]
R. Loos and V. Weispfenning. Applying linear quantifier elimination. Technical report, Wilhelm Schickard-Institut für Informatik, Universität Tübingen, Germany, 1990.

[Nebel, 1989]
B. Nebel. Terminological cycles: Semantics and computational properties. In *Proceedings of the Workshop on Formal Aspects of Semantic Networks*, 1989.

[Patel-Schneider, 1989]
P.F. Patel-Schneider. Undecidability of subsumption in NIKL. *Artificial Intelligence*, 39(2), 1989.

[Schmidt-Schauß and Smolka, 1991]
M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Journal of Artificial Intelligence*, 48(1):1–26, 1991.

[Schmidt-Schauß, 1989]
M. Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In *First International Conference On Principles of Knowledge Representation and Reasoning*, 1989.

[Schmolze, 1989]
J. Schmolze. Terminological knowledge representation systems supporting n-ary terms. In *First International Conference on Principles of Knowledge Representation and Reasoning*, 1989.

[Tarski, 1951]
A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. U. of California Press. Berkley, 1951.

[Weispfenning, 1988]
V. Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5, 1988.

Approximation in Concept Description Languages

Marco Cadoli
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
via Salaria 113, 00198 Roma, Italia
cadoli@assi.ing.uniroma1.it

Marco Schaerf
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
via Salaria 113, 00198 Roma, Italia
schaerf@assi.ing.uniroma1.it

Abstract

One of the main characteristics of logical reasoning is its high computational complexity. Approximation is one possible way to overcome this problem that has recently received significant attention in the AI literature. The approximation method we propose relies on the definition of two sequences of sets converging to the set of strings that represent the reasoning problem. One sequence contains only supersets of the target set, while the other one contains only subsets. From the logical point of view the approximating sets represent either sound or complete forms of reasoning. The approximating sets are carefully designed both from the computational and from the semantical point of view. Our goal in this paper is to use this method for the approximation of reasoning problems typical of *concept description languages*. Concept languages are abstractions for several languages in Computer Science and AI. In this paper we demonstrate our approximation method for the two concept languages $\mathcal{ALE}$ and $\mathcal{ALC}$. The method we propose has both a syntactic and a semantic account.

1 Introduction

One of the main characteristics of logical reasoning is its high computational complexity. Although significant exceptions exist, most of the logical problems which are interesting from the point of view of AI are computationally unfeasible (polynomially intractable) or even undecidable.

Approximation is a technique which is commonly used in many areas of Computer Science to deal with the computational intractability of problems. The major difficulty about introducing approximation in reasoning problems is in that it is very hard to find a measure of the approximation which is not dependent on the particular problem at hand. Nevertheless, some approaches to the approximation of logical problems have recently appeared in the AI literature. Levesque's [18] architecture based on incomplete reasoning, Frisch's [13, 14] limited inference systems, Crawford and Kuiper's [8] access-limited logic, Kautz and Selman's [16, 28] knowledge compilation and Dean and Boddy's [9] anytime algorithms, further investigated by Russell and Zilberstein in [26] and by Ginsberg in [15], are among the most valuable approaches.

The present authors proposed in previous works [5, 7, 6] a general technique for the approximation of complex reasoning problems. A reasoning task is modeled as the problem of deciding whether a string x belongs to a set $\mathcal{D}$ (e. g. the set of satisfiable propositional formulae) or not. The method we defined for approximating a reasoning problem, or equivalently the corresponding set $\mathcal{D}$, relies on the definition of two sequences of sets $\langle \mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_m \rangle$ and $\langle \mathcal{D}^0, \mathcal{D}^1, \ldots, \mathcal{D}^n \rangle$ having the following properties:

- $\mathcal{D}_0 \subseteq \mathcal{D}_1 \subseteq \cdots \subseteq \mathcal{D}_m = \mathcal{D} = \mathcal{D}^n \subseteq \mathcal{D}^{n-1} \subseteq \cdots \subseteq \mathcal{D}^0$;
- the length n, m of the sequences is polynomial with respect to the input of the problem;
- the elements of both sequences are defined by means of a semantics closely related to that of $\mathcal{D}$;
- deciding whether the input string x belongs to $\mathcal{D}^0$ or not (or loosely, deciding membership in $\mathcal{D}^0$) is a polynomial problem; the same holds for $\mathcal{D}_0$. In general deciding membership in $\mathcal{D}^i$ and in $\mathcal{D}_i$ gets exponentially harder as i grows, but it is not harder than deciding membership in $\mathcal{D}$.

The reasoning task is performed in an incremental fashion, by deciding membership in sets $\mathcal{D}_i$ and $\mathcal{D}^j$ of both sequences for increasing indexes i, j, starting with $i = j = 0$. If we prove membership in any $\mathcal{D}_i$, then we have also proved membership in $\mathcal{D}$; on the other hand if we disprove membership in any $\mathcal{D}^i$, then we have also disproved membership in $\mathcal{D}$.

There are clearly two possibilities in using this method: either we are able to solve the reasoning problem in a reasonable amount of time (tipically for small indeces i, j), or i, j get too big and we run out of computing resources. Since the reasoning problems taken into account are polynomially intractable, the latter case will be frequent, therefore it has to be analyzed very carefully.

The whole purpose of our research on approximation is the study of how to give a meaningful answer in this case. We believe that such a meaningful answer has to be grounded on a simple semantics and justified by intuitive arguments. In this way we want to obtain an understandable reasoning system whose precision is arbitrary, but depends on the computational effort that has been spent. The system is realized by means of a stepwise procedure that can be interrupted at any step in such a way that the information obtained so far gives interesting semantic insights. Following ideas of Levesque [17, 18, 19], we grounded the semantics for approximation on a multivalued logic.

In our previous works we approximated several sets of great interest in AI. In [5] we dealt with the set $\{\langle T;\gamma\rangle \mid T,\gamma$ are propositional formulae and $T \models \gamma\}$, in [7] with the set $\{\gamma \mid \gamma$ is a satisfiable formula of the modal system **K**$\}$ and in [6] with the set $\{\langle\Delta;\gamma\rangle \mid \Delta$ is a propositional default theory and γ credulously follows from $\Delta\}$. Our method is to some extent independent on the complexity of the reasoning task. The complexity of the reasoning problems represented by these sets is co-NP-complete, PSPACE-complete and Σ_2^p-complete, respectively. Other basic reasoning tasks of propositional, modal, autoepistemic and nonmonotonic logics have been approximated in the same papers.

Our goal in this paper is to show that our method can be used for the approximation of reasoning problems typical of *concept description languages*. The interest in concept description — also called terminological — languages originated from the study of knowledge representation systems, such as KL-ONE (see [4]). Concept languages are (generally decidable) sublanguages of predicate logic that are abstractions for several languages in Computer Science and Artificial Intelligence. The importance of concept languages in data modeling [4], object oriented data bases [2] and logic programming [1] has been stressed by several authors.

The computational complexity of concept languages has been extensively studied by many authors (see [3, 10, 11, 12, 21, 22, 23, 27]). In particular, it has been shown that reasoning is polynomially intractable in many interesting cases, and that tractability depends on small variations of the expressiveness of the language. A big effort has been spent in the design of "maximally polynomial languages", i. e. polynomial languages such that no expressiveness can be added without losing tractability. Other researchers (see [20, 24]) proposed incomplete or unsound reasoning systems based on non-standard semantics in order to simplify reasoning tasks for concept languages.

In this paper we focus our attention on the languages belonging to the $\mathcal{AL}$- family (see [27] for an overview on the family) and in particular on $\mathcal{ALE}$ and $\mathcal{ALC}$. As shown in [11], these languages are representative of a wide class of concept languages.

The structure of the paper is as follows: in the following section we recall the basic notions on concept languages and introduce multivalued logics for approximation in first-order languages. In Sections 3 and 4 we demonstrate our approximation method for the two languages $\mathcal{ALE}$ and $\mathcal{ALC}$, respectively. Finally, in Section 5 we compare our work with other approaches appeared in the literature and we draw some conclusions. The Appendix contains the proofs of the theorems.

2 Preliminaries

In this section we introduce some technical notions that are necessary for explaining our method for the approximation of reasoning in concept languages.

2.1 Concept Languages

In this subsection we summarize the syntax and semantics of concept description languages.

Concept languages are decidable sublanguages of predicate logic, designed for dealing with *concepts*. A concept is a monadic predicate that can be built up of two kinds of symbols, primitive concepts and roles. Primitive entities can be combined by various language constructors yielding complex concepts. A primitive concept is simply a symbol, like female, that is used to represent a class of individuals all having a common property, like being female. A primitive role is a symbol, like friend, that is used to represent a binary relation among individuals, like friendship. A typical example of language constructor is the universal quantification $\forall$. The symbol $\forall$friend.female is a composite concept that represents the set of individuals having all the friends being female. Other typical language constructors are the existential quantification $\exists$ and the boolean connectives $\sqcap$, $\sqcup$, $\neg$. Another example of composite concept is $\exists$friend.$\neg$female, that represents the set of individuals having at least one friend being not female. In this paper we focus our attention on the two languages $\mathcal{ALE}$ and $\mathcal{ALC}$ belonging to the $\mathcal{AL}$- family (see [27]).

Throughout this section we denote primitive concepts with the letters A, B, concepts (either primitive or composite) with the letters C, D and primitive roles with the letter R.

The syntax of the language $\mathcal{ALC}$ is the following:

$$C, D \longrightarrow \top \mid \bot \mid A \mid \neg A \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$

The special symbols $\top$ and $\bot$ stand for the universal concept and the empty concept, respectively. All the individuals belong to $\top$, while no individual belongs to $\bot$. Although the standard syntax of $\mathcal{ALC}$ allows negation in front of any concept, due to the presence of disjunction we don't lose any generality by allowing negation only in front of primitive concepts. Examples of well-formed $\mathcal{ALC}$ concepts are:

$$(\exists R.A) \sqcap (\exists R.(\neg A \sqcap \forall R.\exists R.A))$$

and

$$\exists R.\exists R.\forall R.(A \sqcup B).$$

$\mathcal{ALE}$ is another interesting language, whose syntax is a restriction of that of $\mathcal{ALC}$. In particular the construct $C \sqcup D$ is not allowed in $\mathcal{ALE}$, therefore the first of the two $\mathcal{ALC}$ concepts above is also an $\mathcal{ALE}$ concept, while the second is not.

The semantics of concept languages is usually given by defining a domain of interpretation U, by assigning to every primitive concept a subset of U and to every primitive role a subset of $U \times U$ and by defining a rule for each constructor. For the purpose of this paper we prefer to define the semantics of $\mathcal{ALE}$ and $\mathcal{ALC}$ by translating any concept C into a first-order sentence $\Gamma(C)$ and interpreting it with the classical semantics of first-order logic. As an example, the $\mathcal{ALC}$ concept $D \equiv (B \sqcap \forall R.A) \sqcup \exists R.\neg B$ is translated into the first-order formula

$$(B(a) \wedge \forall x R(a,x) \rightarrow A(x)) \vee (\exists y R(a,y) \wedge \neg B(y))$$

where a is a constant symbol. In general the translation from a concept C into the corresponding first order formula is obtained by applying the following rewriting rules to C:

$$\begin{aligned}
\Gamma(C) &\longrightarrow \Phi(C,a) \\
\Phi(\top,x) &\longrightarrow \mathit{true} \\
\Phi(\bot,x) &\longrightarrow \mathit{false} \\
\Phi(A,x) &\longrightarrow A(x) \\
\Phi(\neg A,x) &\longrightarrow \neg A(x) \\
\Phi(C \sqcap D,x) &\longrightarrow \Phi(C,x) \wedge \Phi(D,x) \\
\Phi(C \sqcup D,x) &\longrightarrow \Phi(C,x) \vee \Phi(D,x) \\
\Phi(\forall R.C,x) &\longrightarrow \forall y R(x,y) \rightarrow \Phi(C,y) \\
\Phi(\exists R.C,x) &\longrightarrow \exists y R(x,y) \wedge \Phi(C,y)
\end{aligned}$$

where a is a constant symbol, y is a variable symbol and x is either the constant symbol a or a variable symbol. We assume that the new variables introduced by the last two rules are fresh. A concept C is *satisfiable* iff its translation $\Gamma(C)$ is a satisfiable sentence, *unsatisfiable* otherwise. As an example, the concept $\forall R.A$ is satisfiable, while the concept $(\forall R.A) \sqcap \exists R.\neg A$ is unsatisfiable. It is immediate to show that this semantics is equivalent to the more standard semantics for concept languages as found, for example, in [27].

In this paper we are interested in analyzing satisfiability problems, although the computational complexity of other reasoning tasks like subsumption and instance checking has been extensively studied in the literature. The complexity analysis has shown that satisfiability checking is PSPACE-complete for $\mathcal{ALC}$ [27] and co-NP-complete for $\mathcal{ALE}$ [10].

2.2 Multivalued Logics for Approximation

In this subsection we define a generalized notion of interpretation of first-order formulae. The main characteristic of the new form of interpretation relies on the fact that the truth values assigned to an atom α and to its negation $\neg\alpha$ aren't necessarily complementary. In particular we generalize to a first-order language the method we proposed in [5] for the propositional case.

Let $\mathcal{L}$ be a first-order language and $\mathcal{D}$ be the set of all the sentences built on $\mathcal{L}$ which are satisfiable. We are interested only in decidable reasoning tasks, hence we assume some suitable restrictions on $\mathcal{L}$ so that $\mathcal{D}$ is decidable. Our goal is to define two sequences of sets converging to $\mathcal{D}$, in the sense specified in the Introduction. We also want to characterize the elements of those sequences from the semantical point of view. In order to simplify our semantic definitions, it is useful to consider a fixed domain of interpretation for the models of a formula. One simple way to obtain this is to concentrate only on Herbrand interpretations. To this end, we assume that the sentences built on $\mathcal{L}$ are in Skolem Normal Form. We are interested only in studying the satisfiability of sentences, therefore we don't lose generality by appealing to the Herbrand theorem.

We recall that the Herbrand Universe U_h of a formula ϕ is the set of terms that can be built using the constant and function symbols occurring in ϕ. The Herbrand Base B_h of ϕ is the set of ground atoms built applying predicate symbols occurring in ϕ to terms of the Herbrand Universe. A Herbrand interpretation of the language $\mathcal{L}$ used in ϕ is a pair $I = < I^+, I^- >$ of subsets of the Herbrand base B_h which partition B_h. In other words I is built using two rules:

Rule 1 $I^+ \cup I^- = B_h$;

Rule 2 $I^+ \cap I^- = \emptyset$.

The subset I^+ represents the set of atoms α of B_h which are true in I, while I^- represents the set of atoms α of B_h whose negation is true in I. Since there are only two possibilities for an atom, namely it may belong either to I^+ or to I^-, we are entitled to call the partition I a *2-interpretation* of $\mathcal{L}$. A 2-interpretation I satisfies an atom α ($I \models \alpha$) iff $\alpha \in I^+$ and satisfies a negated atom $\neg\alpha$ ($I \models \neg\alpha$) iff $\alpha \in I^-$. The satisfaction of complex sentences is defined using the standard rules for disjunction, conjunction and universal quantification:

∨-rule $I \models \alpha \vee \beta$ iff $I \models \alpha$ or $I \models \beta$;
∧-rule $I \models \alpha \wedge \beta$ iff $I \models \alpha$ and $I \models \beta$;
∀-rule $I \models \forall x.\gamma(x)$ iff $I \models \gamma(t)$ for all $t \in U_h$.

We are not interested in rules for the interpretation of complex negated formulae, since it is always possible to "push" the negation in front of atoms.

The main idea of our semantic notion of approximation is to define interpretation of formulae without using one of **Rule 1**, **Rule 2**. This idea is not completely new. Levesque in [18, 19] defines an interpretation of propositional formulae in which **Rule 2** may not hold[1]. We define a *3-interpretation* of $\mathcal{L}$ to be a pair $I =< I^+, I^- >$ of subsets of B_h such that **Rule 1** holds and **Rule 2** may not hold. In other words it is possible that both an atom and its negation are satisfied by I, hence a formula like $\alpha \wedge \neg\alpha$ could be satisfied by a 3-interpretation. The term 3-interpretation is justified by the fact that there are 3 possibilities for each atom α of B_h: 1) $\alpha \in I^+$, $\alpha \notin I^-$, 2) $\alpha \notin I^+$, $\alpha \in I^-$, 3) $\alpha \in I^+$, $\alpha \in I^-$. Intuitively, in the last case we have a contradictory interpretation of the atom α. Interpretation of complex formulae is defined by means of **∨-rule**, **∧-rule**, **∀-rule**.

The notions of satisfiability and entailment are defined in the usual way for both forms of interpretation. A formula γ is 2-satisfiable [3-satisfiable] if there exists a 2-interpretation [3-interpretation] satisfying it. A formula γ is 2-entailed [3-entailed] by a formula T if all the 2-interpretations [3-interpretations] satisfying T satisfy also γ. The notion of 3-satisfiability is trivial, since every formula is 3-satisfiable. On the other hand 3-entailment is sound and not complete wrt 2-entailment. In the propositional case 3-entailment has an interesting computational property noticed by Levesque in [18, 19]: it can be checked in polynomial time for formulae in conjunctive normal form. As suggested by the same author, 3-entailment can be used as a first approximation of propositional entailment, which is a co-NP-complete problem. Propositional 3-entailment can also be defined proof-theoretically, since it can be modeled by a Hilbert-style calculus without *modus ponens* rule.

In a previous work [5] we generalized the idea of Levesque by reintroducing **Rule 2** in a limited way. In that paper we dealt with the propositional case, now we are going to give the definition for the first-order case. Let S be a subset — even not proper — of the Herbrand base B_h.

Definition 1 (*S*-3-interpretation) *An S-3-interpretation of $\mathcal{L}$ is a pair $I =< I^+, I^- >$ where both I^+ and I^- are subsets of B_h, such that*

Rule 1 $I^+ \cup I^- = B_h$;

Rule 2′ $I^+ \cap I^- \cap S = \emptyset$.

Intuitively, an S-3-interpretation is a 2-interpretation of the atoms of S, while it is a 3-interpretation of the remaining atoms. Notice that, for any S, a 2-interpretation is always an S-3-interpretation, while the latter is always a 3-interpretation. Satisfaction of complex sentences is defined by means of **∨-rule**, **∧-rule**, **∀-rule**. Notice also that if $S \neq \emptyset$, then formulae which are S-3-unsatisfiable exist. In particular S-3-unsatisfiability of a formula always implies its 2-unsatisfiability, since the set $\mathcal{D}$ of 2-satisfiable formulae is a subset of the set of S-3-satisfiable formulae, for any $S \subseteq B_h$. More precisely, if $S \subseteq S' \subseteq B_h$, then S-3-unsatisfiability implies S'-3-unsatisfiability. Using this characteristic of S-3-satisfiability we can define a sequence of sets converging to $\mathcal{D}$. As an example we used this method in [5] for approximating the set $\mathcal{D}_{prop}$ of 2-satisfiable propositional formulae. In the propositional case S is a set of boolean variables and S-3-satisfiability of a formula T can be checked in time $O(|T| \cdot 2^{|S|})$.

The incremental solution of a satisfiability problem, i. e. to decide whether a sentence belongs to the set $\mathcal{D}$, is defined via a sequence $\langle \mathcal{D}^0, \mathcal{D}^1, \ldots, \mathcal{D}^n \rangle$, where $\mathcal{D}^n$ is $\mathcal{D}$, $\mathcal{D}^i$ is the set of S_i-3-satisfiable sentences, $\langle S_0, S_1, \ldots, S_n \rangle$ is an increasing sequence of subsets of B_h and S_n is B_h. The sequence of sets defined in this way fulfils the desiderata about an approximation schema we listed in the Introduction. In the following sections we give concrete examples of choice of the sequence $\langle S_0, S_1, \ldots, S_n \rangle$, in the context of the languages $\mathcal{ALE}$ and $\mathcal{ALC}$.

So far we were concerned with the definition of a sequence of supersets of $\mathcal{D}$ converging to it. We want now to give a semantic definition for a converging sequence of subsets of $\mathcal{D}$. To this end we introduce the definition of S-1-interpretation, which is dual to S-3-interpretation and modifies **Rule 1**.

Definition 2 (*S*-1-interpretation) *An S-1-interpretation of $\mathcal{L}$ is a pair $I =< I^+, I^- >$ where both I^+ and I^- are subsets of B_h, such that*

Rule 1′ $I^+ \cup I^- = S$;

Rule 2 $I^+ \cap I^- = \emptyset$.

An S-1-interpretation is a 2-interpretation as long as the atoms of S are concerned. There is only one possibility for an atom α not in S: $\alpha \notin I^+$, $\alpha \notin I^-$. This justifies the term S-1-satisfiability. In [5] we used this definition for approximating the set $\mathcal{D}_{prop}$. In the propositional case S-1-satisfiability implies 2-satisfiability. More precisely if $S \subseteq S' \subseteq B_h$, then S-1-satisfiability implies S'-1-satisfiability. The property of a formula T of being S-1-satisfiabile can be decided in time $O(|T| \cdot 2^{|S|})$.

[1] Loosely speaking, the Herbrand Base of a propositional formula is the set of the boolean variables occurring in it.

Taking into account the interpretation of first-order sentences, we define $Terms(S)$ to be the set of all the terms that can be generated by using the functions and constants which appear in atoms of S. Notice that this is in general a subset of the Herbrand Universe. We define S-1-interpretation by means of $\vee$-**rule**, $\wedge$-**rule**, and the following

$\forall'$-**rule** $I \models \forall x.\gamma(x)$ iff $I \models \gamma(t)$ for all $t \in Terms(S)$.

As we show in the following sections, we are not interested in sets S containing all the ground instances of a predicate symbol. S-1-interpretation with the $\forall$-**rule** would be trivial, since no sentence can be satisfied. The intuition behind the $\forall'$-**rule** is that we are ignoring objects that are not in the intended domain of interpretation.

3 Approximation in $\mathcal{ALE}$

In this section we define a method for approximating the task of deciding satisfiability of an $\mathcal{ALE}$ concept. The method is based on a syntactic manipulation of concepts that simplifies the task of checking their satisfiability. The syntactic manipulation is given a precise semantics in terms of S-1- and S-3-interpretations.

The method we are going to present is based on the idea of approximating an $\mathcal{ALE}$ concept by means of two sequences of "simpler" $\mathcal{ALE}$ concepts. There are two ways in which a concept can be simpler than another one: a concept can be approximated either by a ***weaker*** concept or by a ***stronger*** one. A concept D is weaker than C if it represents a class with weaker properties, i. e. a less specific class. On the other hand, stronger concepts represent more specific classes. Both kinds of approximated concepts carry interesting information. In fact, if we can prove that a weaker concept is unsatisfiable, then the unsatisfiability of C is also proved. Proving the satisfiability of a stronger concept implies the satisfiability of C. One of the two sequences defining the approximation contains only weaker concepts. It starts with a very rough approximation and is improved in a stepwise process, giving "stronger and stronger" approximations and eventually converging to the original concept. The second sequence is dual, and contains only stronger concepts.

As an example, let's consider an unsatisfiable $\mathcal{ALE}$ concept, called *dummy* in the following

$$(\exists \text{friend.tall}) \sqcap \forall \text{friend.}((\forall \text{friend.doctor}) \sqcap \exists \text{friend.}\neg \text{doctor}).$$

It denotes the (empty) set of the individuals having at least one friend tall and all the friends having all the friends doctor and at least one friend having a friend who is not a doctor.

For obtaining concepts approximating *dummy* we syntactically simplify it by substituting complex subconcepts with simpler ones, where a subconcept D of an $\mathcal{ALE}$ concept C is a substring of C being an $\mathcal{ALE}$ concept. The *depth* of D is the number of universal quantifiers occurring in C and having D in their scope. For example, the depth of the subconcept $\exists$friend.$\neg$doctor of *dummy* is 1, while the depth of $\exists$friend.tall is 0. We define the depth of a concept to be the maximum depth of its existentially quantified subconcepts. The depth of *dummy* is 1.

Using the notion of depth we define the sequence of weaker approximated concepts. The i-th weaker concept is obtained by replacing every existentially quantified subconcept of depth greater or equal than i with the primitive concept $\top$. As an example, taking into account *dummy* we obtain the sequence of concepts:

- $\top \sqcap \forall \text{friend.}((\forall \text{friend.doctor}) \sqcap \top)$
- $(\exists \text{friend.tall}) \sqcap \forall \text{friend.}((\forall \text{friend.doctor}) \sqcap \top)$
- *dummy*.

The elements of the sequence are denoted as $dummy_0^\top$, $dummy_1^\top$, $dummy_2^\top$, respectively and are all unsatisfiable. Notice that the first two concepts are both satisfiable, while the third one is unsatisfiable. The sequence of the stronger concepts is obtained by substituting $\bot$ instead of $\top$. Its elements are denoted as $dummy_0^\bot$, $dummy_1^\bot$, $dummy_2^\bot$, respectively.

Formally, we associate to an $\mathcal{ALE}$ concept C of depth n two distinct sequences $\sigma^\top = C_0^\top, \ldots, C_{n+1}^\top$ and $\sigma^\bot = C_0^\bot, \ldots, C_{n+1}^\bot$ of $\mathcal{ALE}$ concepts. For each i ($0 \leq i \leq n$), every concept $C_i^\top$ [$C_i^\bot$] of the sequence $\sigma^\top$ [$\sigma^\bot$] is obtained from C by substituting every existentially quantified subconcept of C which is in the scope of at least i universal quantifiers with the concept $\top$ [$\bot$]. Moreover $C_{n+1}^\top = C_{n+1}^\bot = C$. Notice that, for each i ($1 \leq i \leq n+1$) the depth of $C_i^\top$ [$C_i^\bot$] is strictly less than i.

The semantics of the approximation will be addressed later on, but at this point we would like to make some intuitive considerations on the simplified concepts. As we noticed in Subsection 2.1, proving the satisfiability of a concept C is equivalent to proving the existence of an object a having the property represented by the first-order formula $\Gamma(C)$. If existentially quantified subconcepts of depth 0 occur in C, then it is necessary to consider objects related to a. As an example, when we deal with the concept *dummy* we need to consider the existence of a friend of a being tall, because of the subconcept $\exists$friend.tall of C. Let's call c this friend. Deeper existentially quantified subconcepts may also contribute to the generation of objects that have to be considered. Continuing our example, the subconcept $\exists$friend.$\neg$doctor of *dummy* makes it necessary to take into account the existence of a friend of c — let's call it $f(c)$ — not being a doctor. Intuitively, focusing on

simplified concepts means ignoring the properties of objects which are "far away" from a. As an example, considering $dummy_1^\top$ frees us from taking into account the object $f(c)$, while when we consider $dummy_0^\top$ we don't even have to deal with the object c. This intuition will be clarified when we will analyze the semantics of the concepts in the approximating sequences.

Now we want to give a couple of important properties of the sequences $\sigma^\top$ and $\sigma^\bot$, that are useful for defining our approximation schema.

Theorem 1 (monotonicity of $\sigma^\top$ and $\sigma^\bot$) *For each i $(0 \leq i \leq n+1)$, if $C_i^\top$ is unsatisfiable then $C_j^\top$ is unsatisfiable for all $j \geq i$, hence C is unsatisfiable.*

For each i $(0 \leq i \leq n+1)$, if $C_i^\bot$ is satisfiable then $C_j^\bot$ is satisfiable for all $j \geq i$, hence C is satisfiable.

The proofs of all theorems are in the appendix.

Observation 2 (convergence of $\sigma^\top$ and $\sigma^\bot$) *If C is unsatisfiable then there exists an i $(0 \leq i \leq n+1)$ such that $C_i^\top$ is unsatisfiable. If C is satisfiable then there exists an i $(0 \leq i \leq n+1)$ such that $C_i^\bot$ is satisfiable.*

The above properties suggest a method for deciding in an incremental fashion the satisfiability of an $\mathcal{ALE}$ concept C. We may start by deciding the satisfiability of $C_0^\top$; if $C_0^\top$ is unsatisfiable, then by Theorem 1 we are guaranteed that C is unsatisfiable as well. Analogously, if $C_0^\bot$ is satisfiable, then we know that C is satisfiable. If neither of the two cases happens, then we decide the satisfiability of $C_1^\top$ and $C_1^\bot$, and so on. Clearly we have a definite answer as soon as we prove either unsatisfiability of some $C_i^\top$ or satisfiability of some $C_i^\bot$. Referring to the notation defined in the Introduction, we approximate the set $\mathcal{D}$ of satisfiable $\mathcal{ALE}$ concepts of depth n by means of two sequences of sets $\langle \mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{n+1} \rangle$ and $\langle \mathcal{D}^0, \mathcal{D}^1, \ldots, \mathcal{D}^{n+1} \rangle$, where $\mathcal{D}_i$ $[\mathcal{D}^i]$ is the set of $\mathcal{ALE}$ concepts C whose stronger [weaker] form $C_i^\bot$ $[C_i^\top]$ is satisfiable.

We want now to make some considerations on the computational cost of deciding satisfiability of an $\mathcal{ALE}$ concept in such an incremental fashion. Donini et al. show in [10] that the problem of checking the satisfiability of an $\mathcal{ALE}$ concept whose depth is linear in its length is co-NP-complete. In the same paper it is shown that the satisfiability of any $\mathcal{ALE}$ concept having depth m can be checked in time proportional to $l \cdot 2^m$, where l is the length of the concept. In other words, the nesting of existential and universal quantifiers is the crucial measure of the complexity of satisfiability checking. Our method for checking satisfiability of an $\mathcal{ALE}$ concept C considers simplified versions of C of increasing depth and may use existing algorithms for checking their satisfiability. The complexity of the whole method is $O(l^2 \cdot 2^m)$ even if satisfiability cannot be decided until the unsimplified concept C is taken into account. In the worst case the complexity of our method is therefore comparable to that of the existing algorithms. Since the problem is co-NP-complete, we don't expect any algorithm to be significantly better than ours in the worst case.

We are now interested in giving a clear semantics to our approximation schema. This is particularly important, since as we stressed earlier it is not always possible to obtain in a reasonable amount of time a definite answer to the problem of checking the satisfiability of an $\mathcal{ALE}$ concept. Therefore the meaning of each step of the approximation should be very clear, since in general we can afford only an approximate solution.

In order to give a semantic account to the approximations of $\mathcal{ALE}$ concepts, we are now going to use the notions of S-3- and S-1-interpretation introduced in Subsection 2.2. As we noticed earlier, S-3-interpretations lead to a complete (and in general unsound) definition of satisfiability. Analogously, Theorem 1 shows that satisfiability of the concepts of the sequence $\sigma^\top$ is complete and unsound wrt the satisfiability of the original concept C. Our goal is to show that 2-satisfiability (or simply, satisfiability) of each concept $C_i^\top$ is equivalent to S_i-3-satisfiability of C for a suitable subset S_i of the Herbrand Base of the Skolem Normal Form $SNF(\Gamma(C))$ of $\Gamma(C)$. An analogous result will be obtained for a concept $C_i^\bot$ and S_i-1-satisfiability of C.

As we said in Subsection 2.1, the semantics of an $\mathcal{ALE}$ concept C can be defined in terms of a first-order formula $\Gamma(C)$. Since we are only interested in satisfiability properties, we take into account the Skolem Normal Form $SNF(\Gamma(C))$ of $\Gamma(C)$. The Herbrand Universe of $SNF(\Gamma(C))$ can be stratified in a very simple way by taking into account the increasing complexity of its terms. This stratification delivers another simple stratification on the Herbrand Base. More precisely, let U_h be the Herbrand Universe of $SNF(\Gamma(C))$. This universe can be stratified with the following policy: Let U_0 be $\{a\}$ and U_i be the set of all the terms which can be formed using a, Skolem constants of U_h and functions of U_h of arity strictly less than i. Clearly it holds that $U_0 \subseteq U_1 \subseteq \ldots \subseteq U_{n+1} = U_h$. Notice that a function of arity i comes from the skolemization of an existential quantifier of depth i. This stratification of the Herbrand Universe of $SNF(\Gamma(C))$ induces a stratification $S_0 \subseteq S_1 \subseteq \ldots \subseteq S_{n+1} = B_h$ on its Herbrand Base H_b defined in the following way:

$$\begin{aligned} S_i \;=\; & \{A(t) \mid A \text{ is a primitive concept and } t \in U_i\} \\ & \cup \{R(t_1, t_2) \mid R \text{ is a role and } t_1, t_2 \in U_i\}. \end{aligned}$$

A clarifying example is in order. Let's consider again the concept *dummy*

$$(\exists \text{friend.tall}) \sqcap$$
$$\forall \text{friend.}((\forall \text{friend.doctor}) \sqcap \exists \text{friend.}\neg \text{doctor})$$

and the related first-order formula $\Gamma(dummy)$, in which we use obvious abbreviations for the predicate symbols

$$(\exists x F(a,x) \wedge T(x)) \wedge \forall y F(a,y) \rightarrow$$
$$(\forall z(F(y,z) \rightarrow D(z)) \wedge (\exists u F(y,u) \wedge \neg D(u)))$$

The following formula is the Skolem Normal Form $SNF(\Gamma(dummy))$ of $\Gamma(dummy)$

$$F(a,c) \wedge T(c) \wedge \forall y(F(a,y) \rightarrow$$
$$(\forall z(F(y,z) \rightarrow D(z)) \wedge F(y,f(y)) \wedge \neg D(f(y))))$$

In $SNF(\Gamma(dummy))$, c is a new constant symbol that replaces the variable x, while $f()$ is a new function symbol —having arity 1— replacing the variable u. The Herbrand Universe U_h of $SNF(\Gamma(dummy))$ is the set of all the terms that can be obtained from a, c and $f()$. The Universe U_h is stratified into the three sets $U_0 = \{a\}, U_1 = \{a,c\}, U_2 = \{a,c,f(a),f(c),f(f(a)),\ldots\}$.

Notice that the S_i turns out to be equal (up to Skolem functions and constants renaming) to the Herbrand Base of $SNF(\Gamma(C_i^\top))$.

As we saw before the concept *dummy* is unsatisfiable. We noticed informally that its unsatisfiability can only be proven by taking into account the properties of the friends of the friends of a: All the members of this class should be doctors, while one of them is not a doctor. As a consequence, *dummy* is unsatisfiable, but its simplified versions $dummy_0^\top$ and $dummy_1^\top$ are both satisfiable. As a semantic counterpart, we notice that $SNF(\Gamma(dummy))$ is unsatisfiable and this follows from the existence of a term of the form $f(c)$ which denotes the friend of the friend c of a. Actually $SNF(\Gamma(dummy))$ is both S_0-3- and S_1-3- satisfiable, where a concept C is S_i-3-satisfiable if $SNF(\Gamma(C))$ is S-3-satisfiable with $S = S_i$. This can be easily shown by noticing that the atom $D(f(c))$ does not belong to S_1, hence it is possible to define a S_1-3-interpretation $I = \langle I^+, I^- \rangle$ such that $D(f(c)) \in I^+, D(f(c)) \in I^-$. This kind of interpretation "hides" the reason of inconsistency of the concept *dummy*, which is therefore S_1-3-satisfiable. This argument obviously holds also for the stratum S_0.

The above example shows that it is possible to relate S_i-3-satisfiability of a concept C of $\mathcal{ALE}$ to satisfiability of the i-th element $C_i^\top$ of the sequence $\sigma^\top$. The following theorem formalizes this result.

Theorem 3 *For all i $(0 \leq i \leq n+1)$ C is S_i-3-satisfiable iff $C_i^\top$ is satisfiable.*

What this theorem says is that our approximation schema based on the analysis of syntactically simplified concepts readily corresponds to the semantic idea of focusing on subsets of the Herbrand Base which are defined by a simpler universe. This characterization of the approximation process could not be obtained if we were to use the more standard semantics for $\mathcal{AL}$-languages (see the Appendix) based on extension functions, since we could not introduce the notion of complexity of terms.

The above correspondence between S_i-3-satisfiability of a concept C of $\mathcal{ALE}$ and satisfiability of the i-th element $C_i^\top$ of the sequence $\sigma^\top$ can be easily extended to the dual case of S_i-1-satisfiability. Similarly to the previous case, it is possible to show that the approximation of the satisfiability of a concept C through the sequence $\sigma^\bot$ can be interpreted as S_i-1-satisfiability of the translated concept $SNF(\Gamma(C))$. This follows from the equivalence of the Herbrand Base of $SNF(\Gamma(C_i^\bot))$ and the Herbrand Base of $SNF(\Gamma(C_i^\top))$.

Theorem 4 *For all i $(0 \leq i \leq n+1)$ C is S_i-1-satisfiable iff $C_i^\bot$ is satisfiable.*

Notice that the difference between S_i-1- and S_i-3-interpretations relies on the interpretation of atoms not belonging to the stratum S_i. As an example, in each S_1-1-interpretations I of the concept *dummy* it holds $D(f(c)) \notin I^+, D(f(c)) \notin I^-$.

We want to conclude this section by making some considerations about the choice of the subsets S_i of the Herbrand Base of $SNF(\Gamma(C_i^\top))$ defining our approximation schema. Donini et al. show in [10] that the number of primitive concepts and roles used is not a source of complexity in $\mathcal{ALE}$. More precisely, deciding the satisfiability of an $\mathcal{ALE}$ concept C is a co-NP-complete problem even if a single primitive role and no primitive concepts but $\top$ occur in C. This fact has an important impact in the choice of an approximation schema for $\mathcal{ALE}$ concepts. In particular it shows that, if the subset S of the Herbrand base contains all the ground instances of a single atomic role, then deciding S-3-satisfiability of an $\mathcal{ALE}$ concept is still co-NP-complete. Therefore the sets S defining the approximation schema must be designed so that the real source of complexity — which is the depth of concepts — is addressed.

4 Approximation in $\mathcal{ALC}$

In this section we define a method for approximating the task of deciding satisfiability of an $\mathcal{ALC}$ concept. Like the method illustrated in the previous section, also this one is based on a syntactic manipulation of concepts that is interpreted in terms of S-1- and S-3-satisfiability.

A first obvious question is: can we plainly use the method introduced in Section 3 for the purpose of approximating $\mathcal{ALC}$ concepts? We notice that $\mathcal{ALC}$ without existential quantifiers has at least the expressiveness of propositional calculus. As a consequence deciding satisfiability of $\mathcal{ALC}$ concepts without existential quantifiers is an NP-hard problem. This implies that the method defined for the approximation of an $\mathcal{ALE}$ concept leads — when used for $\mathcal{ALC}$ concepts — to polynomially intractable problems from the very first step of the approximation. In our opinion an effective method for the approximation of $\mathcal{ALC}$ should address both sources of complexity of this language (see [11]): the complexity of existentials, also present in $\mathcal{ALE}$, and that of disjunction, present in the existential-free fragment of $\mathcal{ALC}$.

The source of complexity of disjunction is also present in the propositional calculus. The method we proposed in [5] for approximating propositional satisfiability (see also Subsection 2.2) relies on the use of S-1- and S-3-satisfiability, in which sets S containing more and more boolean variables are taken into account. This corresponds to a stratification $S_0 \subseteq S_1 \subseteq \ldots \subseteq S_{n+1} = B_h$ of the Herbrand Base B_h, in which each stratum contains all the ground instances of a predicate symbol. As we noticed at the end of the previous section, such a stratification cannot be used for the approximation of $\mathcal{ALE}$, which must instead be based on the complexity of the terms of the Herbrand Universe. It is therefore natural to combine both ideas for the approximation of $\mathcal{ALC}$.

Let's start again with a concrete example and consider the following $\mathcal{ALC}$ concept:

$$D \equiv (A \sqcap \neg A) \sqcup ((\forall R.A) \sqcap \exists R.\neg A)$$

D is unsatisfiable, because it is the union of two unsatisfiable $\mathcal{ALE}$ concepts. D can be approximated either by substituting the existentially quantified subconcept $\exists R.\neg A$ with $\top$ or by replacing the primitive concept A and its negation $\neg A$ with $\top$. Moreover both methods can be combined. Each of the following $\mathcal{ALC}$ concepts is weaker than D and satisfiable:

$$\begin{aligned} D^{\top}_{\emptyset,0} &\equiv (\top \sqcap \top) \sqcup (\forall R.\top \sqcap \top) \\ D^{\top}_{\{A\},0} &\equiv (A \sqcap \neg A) \sqcup (\forall R.A \sqcap \top) \\ D^{\top}_{\emptyset,1} &\equiv (\top \sqcap \top) \sqcup (\forall R.\top \sqcap \exists R.\top) \end{aligned}$$

The first subscript denotes the set of primitive concepts that are substituted by $\top$, while the second one denotes the depth of the approximation. In general, given an $\mathcal{ALC}$ concept C of depth n built on the set $\mathcal{A}$ of primitive concepts, a set $P \subseteq \mathcal{A}$ and an index $0 \leq i \leq n+1$, we denote with the symbol $C^{\top}_{P,i}$ the $\mathcal{ALC}$ concept obtained from C by means of the following rules:

1. substitute each (positive or negative) occurrence of a primitive concept not in P with the concept $\top$, thus obtaining the concept C';
2. substitute every existentially quantified subconcept of C' which is in the scope of at least i universal quantifiers with the concept $\top$.

The concept $C^{\bot}_{P,i}$ is obtained by substituting $\bot$ instead of $\top$. Let P_1, P_2 be two subsets of $\mathcal{A}$ and let i_1, i_2 be two indexes such that $0 \leq i_1, i_2 \leq n+1$. Let C be an $\mathcal{ALC}$ concept and $C_1 \equiv C^{\top}_{P_1,i_1}$, $C_2 \equiv C^{\top}_{P_2,i_2}$ two approximations of C. We say that $C_1 \preceq C_2$ holds iff both $P_1 \subseteq P_2$ and $i_1 \leq i_2$ hold. We define the relation $\preceq$ only between pairs of weaker approximations and pairs of stronger ones. This relation is a partial order in the set of the approximations of an $\mathcal{ALC}$ concept. The following are two interesting properties of this relation:

Theorem 5 (monotonicity) *Let P be a subset of $\mathcal{A}$ and i be an index such that $0 \leq i \leq n+1$. If $C^{\top}_{P,i}$ is unsatisfiable then any subconcept D of C such that $C^{\top}_{P,i} \preceq D$ is unsatisfiable. If $C^{\bot}_{P,i}$ is satisfiable then any subconcept D of C such that $C^{\bot}_{P,i} \preceq D$ is satisfiable.*

Observation 6 (convergence) *If C is unsatisfiable, then there exists a subset P of $\mathcal{A}$ and an index i ($0 \leq i \leq n+1$) such that $C^{\top}_{P,i}$ is unsatisfiable. If C is satisfiable, then there exists a subset P of $\mathcal{A}$ and an index i ($0 \leq i \leq n+1$) such that $C^{\bot}_{P,i}$ is satisfiable.*

The above properties are analogous to Theorem 1 and Observation 2 for $\mathcal{ALE}$. Following the argument of Section 3, we say that any $\mathcal{ALC}$ concept C can be approximated by means of an increasing (wrt $\preceq$) sequence of weaker concepts $C^{\top}_{P,i}$. This corresponds to a weak approximation of C. A strong approximation can be obtained by considering an increasing sequence of concepts $C^{\bot}_{P,i}$.

It can be shown that satisfiability checking of a subconcept $C^{\top}_{P,i}$ or $C^{\bot}_{P,i}$ can be done in time proportional to $2^{|P| \cdot i}$. This property entitles us to perform an argument similar to that of Section 3, in order to show that our method for checking satisfiability of $\mathcal{ALC}$ concepts in an incremental fashion has a complexity comparable to the standard algorithms (see [27]).

From the semantical point of view it is possible to characterize this form of approximation in terms of S-3- and S-1-interpretations. Let C be an $\mathcal{ALC}$ concept and n be its depth — defined exactly as in the case of $\mathcal{ALE}$ concepts. The stratification $U_0 \subseteq U_1 \subseteq \ldots \subseteq U_{n+1} = U_h$ of the Herbrand Universe U_h of $SNF(\Gamma(C))$ is defined as in the previous section. We define the subset $S_{P,i}$ of the Herbrand Base H_b of $SNF(\Gamma(C))$ to be the following set:

$$\{A(t) \mid A \text{ is a primitive concept in } P \text{ and } t \in U_i\}$$
$$\cup \{R(t_1, t_2) \mid R \text{ is a role and } t_1, t_2 \in U_i\}$$

We say that C is $S_{P,i}$-3-satisfiable iff it is S-3-satisfiable for $S = S_{P,i}$. The intuition behind this definition is that we are confining our attention only to a subset of the set S_i defined in Section 3. In particular we want a unary atom $A(t)$ to be in $S_{P,i}$ only if the concept A belongs to the set of privileged primitive concepts P. In this way we are limiting both sources of potential complexity: disjunction and existential quantification. Contradiction cannot arise from roles, since $\mathcal{ALC}$ does not support negation on roles. In an analogous way we define $S_{P,i}$-3-satisfiability. The following properties formalize the relations between syntax and semantics:

Theorem 7 *For all i $(0 \leq i \leq n+1)$ and $P \subseteq \mathcal{A}$ the concept C is $S_{P,i}$-3-satisfiable iff $C^{\top}_{P,i}$ is satisfiable.*

Theorem 8 *For all i $(0 \leq i \leq n+1)$ and $P \subseteq \mathcal{A}$ the concept C is $S_{P,i}$-1-satisfiable iff $C^{\bot}_{P,i}$ is satisfiable.*

We conclude this section by noticing that the method we proposed for the approximation of $\mathcal{ALC}$ concepts is in some sense underspecified, since we gave no criteria for choosing subsets P of the set of primitive concepts. This topic deserves future research: precise criteria and heuristics have to be developed.

5 Related work and conclusions

In this paper we have shown a method for the approximation of satisfiability problems in the concept description languages $\mathcal{ALE}$ and $\mathcal{ALC}$. The method is based on a simple semantics defined on top of a multivalued logic, but it has also an intuitive syntactic characterization. This is not, however, the first time that non-classical semantics are used in order to deal with the high complexity of reasoning.

The idea of gaining tractability by using weaker semantics and inference procedures has already been used by several researchers. The first ones have probably been Levesque [17], with his work on implicit and explicit belief, and Frisch [13, 14] with his work on inference without chaining. Both works define reasoners that are sound and incomplete wrt classical semantics but which are completely characterized by a non-classical one. Moreover, the entailment relations in these semantics can be computed in polynomial time. Patel-Schneider has generalized the work of Levesque in two directions. In [25] he presents a decidable first-order logic based on a simple type of first-order relevance logic. In [24] the same author devises a polynomially tractable terminological logic based on a 4-valued semantics. This is the first use of weaker semantics for concept languages.

As pointed out by the same authors, these systems successfully managed to lower the complexity of the inference tasks, they are far too weak in sanctioning conclusions to be useful in many situations. They also argued that these weak semantics are the building block on top of which more inferential capabilities should be added, without losing tractability. This is exactly the direction of research we are pursuing in our work. What we have shown is that 1) a very weak semantics can be used as the starting step of high complexity decision problems and 2) on top of this we can define increasingly more complete procedures that are based on stronger semantics. Moreover, we have also presented a dual semantics being stronger than the classical one. What we gained with our approach is the ability to characterize with a precise semantics a wide class of incomplete or unsound inference procedures.

The idea of using both sound and complete approximations has also been used by Kautz and Selman in [16, 28]. They apply this idea to concept languages in [28], proposing to simplify syntactically concepts so that inference is computationally simpler. As an example, they propose to approximate a concept of the concept language $\mathcal{FL}$ with the best pair of its approximating $\mathcal{FL}^-$ concepts. Reasoning in $\mathcal{FL}$ is NP-hard, while it is polyomial in $\mathcal{FL}^-$ (see [3]). Kautz and Selman do not take into account the issue of refining the approximation, which means that their approximating concepts cannot be improved like in our framework.

Acknowledgements

We want to thank Maurizio Lenzerini, Francesco M. Donini and Andrea Schaerf for interesting discussions on concept logics. We also thank an anonymous referee for his useful comments on the presentation of the paper and for pointing out the references [13, 14, 24, 25]. This work has been supported by the ESPRIT Basic Research Action N.3012-COMPULOG and by the Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of the CNR (Italian Research Council). The second author is supported by a fellowship from CNR. The first author acknowledges John McCarthy and Carolyn Talcott for their hospitality at the Computer Science Department of the Stanford University, where part of this research has been developed.

References

[1] H. Aït-Kaci and R. Nasr. Login: a logic programming language with built-in inheritance. *Journal of Logic Programming*, 3:185–215, 1986.

[2] C. Beeri. Data models and languages for databases. In *Proceedings of the International Conference on Database Theory (ICDT-88)*, pages 19–40, 1988.

[3] R. J. Brachman and H. J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI-84)*, pages 34–37, 1984.

[4] R. J. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9:171–216, 1985.

[5] M. Cadoli and M. Schaerf. Approximate entailment. In *Trends in Artificial Intelligence: Proceedings of the 2nd Conference of the Italian Association for Artificial Intelligence*, number 549 in Lecture Notes In Artificial Intelligence, pages 68–77. Springer-Verlag, 1991.

[6] M. Cadoli and M. Schaerf. Approximate inference in default reasoning and circumscription. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, August 1992. In press. Extended version presented at the 4th International Workshop on Nonmonotonic Reasoning, Vermont, May 1992. Notes edited by Kautz H. and Etherington D.

[7] M. Cadoli and M. Schaerf. Approximate reasoning and non-omniscient agents. In *Proceedings of the Fourth Conference on Theoretical Aspects of Reasoning about Knowledge (TARK-92)*, pages 169–183, March 1992.

[8] J. M. Crawford and B. Kuipers. Towards a theory of access limited reasoning. In *Proceedings of the First International Conference on the Principles of Knowledge Representation and Reasoning (KR-89)*, pages 67–78, 1989.

[9] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 49–54, 1988.

[10] F. M. Donini, B. Hollunder, M. Lenzerini, A. Marchetti Spaccamela, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence Journal*, 53:309–327, 1992.

[11] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, pages 151–162. Morgan Kaufmann, 1991.

[12] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proceedings of the Twelth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 458–463, 1991.

[13] A. M. Frisch. Using model theory to specify AI programs. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 148–154, 1985.

[14] A. M. Frisch. Inference without chaining. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 515–519, 1987.

[15] M. L. Ginsberg. Computational considerations in reasoning about action. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, pages 250–261, 1991.

[16] H. A. Kautz and B. Selman. A general framework for knowledge compilation. In *Proc. of International Workshop on Processing Declarative Knowledge (PDK-91)*, 1991.

[17] H. J. Levesque. A logic of implicit and explicit belief. In *Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI-84)*, pages 198–202, 1984.

[18] H. J. Levesque. Logic and the complexity of reasoning. *Journal of Philosophical Logic*, 17:355–389, 1988.

[19] H. J. Levesque. A knowledge-level account of abduction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1061–1067, 1989.

[20] R. M. MacGregor. Inside the LOOM desscription classifier. *SIGART Bullettin*, 2:88–92, 1991.

[21] B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence Journal*, 34(3):371–383, 1988.

[22] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence Journal*, 43:235–249, 1990.

[23] B. Nebel and G. Smolka. Representation and reasoning with attributive descriptions. In K.H. Bläsius, U. Hedtstück, and C.-R. Rollinger, editors, *Sorts and Types in Artificial Intelligence*, number 422 in Lecture Notes In Artificial Intelligence, pages 112–139. Springer Verlag, 1990.

[24] P. F. Patel-Schneider. A four-valued semantics for terminological logic. *Artificial Intelligence Journal*, 38:319–351, 1989.

[25] P. F. Patel-Schneider. A decidable first-order logic for knowledge representation. *Journal of Automated Reasoning*, 6:361–388, 1990.

[26] S. J. Russell and S. Zilberstein. Composing real-time systems. In *Proceedings of the Twelth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 212–217, 1991.

[27] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence Journal*, 48(1):1–26, 1991.

[28] B. Selman and H. A. Kautz. Knowledge compilation using Horn approximations. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 904–909, 1991.

Appendix: proofs

Some of the proofs given in the following use the standard semantics of concept languages based on extension functions. Now we briefly recall this semantics (for more details see [27].

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the domain) and a function $\cdot^{\mathcal{I}}$ that maps every concept to a subset of $\Delta^{\mathcal{I}}$ and every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, $\bot^{\mathcal{I}} = \emptyset$, $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$, $(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$, and $(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$.

A concept C is satisfiable if and only if there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}}$ is non empty. We say C is subsumed by D ($C \sqsubseteq D$) if for every interpretation $\mathcal{I}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

In order to prove some of the results of the paper we need the following lemma. Let C be an $\mathcal{ALC}$ concept and D one of its subconcepts, we denote with $C(D/G)$ the concept obtained by replacing every occurrence of D in C with the concept G.

Lemma 9 *Let C be an $\mathcal{ALC}$ concept and D one of its subconcepts which is not in the scope of any $\neg$ operators. Let G be another $\mathcal{ALC}$ concept. If $D \sqsubseteq G$ then $C(D/G)$ is satisfiable if C is satisfiable. On the other hand, if $G \sqsubseteq D$ then C is satisfiable if $C(D/G)$ is satisfiable.*

PROOF: The proof is done by induction on the structure of C. In the base case we have that $C = D$, thus $C(D/G) = G$ and $D \sqsubseteq G$ then satisfiability of C implies satisfiability of $C(D/G)$. In the general case we have to show that all the language constructors except negation preserve this property. In particular $D \sqsubseteq G$ implies $C' \sqcap D \sqsubseteq C' \sqcap G$, $C' \sqcup D \sqsubseteq C' \sqcup G$, $\exists R.D \sqsubseteq \exists R.G$ and $\forall R.D \sqsubseteq \forall R.G$. The proof is straightforward for all cases. A dual argument holds for the case of $G \sqsubseteq D$. □

PROOF of Theorem 1:
it follows from Lemma 9. In on case we are replacing subconcepts of C with $\top$ and it is always the case that a concept is subsumed by $\top$. Furthermore, even if $\mathcal{ALE}$ allows negation, we never replace subconcepts which are under the scope of $\neg$ operator so the proof of Lemma 9 still holds. In the other case, in which we replace subconcepts of C with $\bot$, the other part of Lemma 9 is used. □

Dealing with an S-3-interpretation M, when an atom $\alpha \in B_h \setminus S$ belongs both to M^+ and to M^-, we say that α is mapped into *inconsistent*.

PROOF of Theorem 3:
(only if part) Suppose that $C_i^{\top}$ is satisfiable, i.e. that $SNF(\Gamma(C_i^{\top}))$ is satisfiable. Let $M = \langle M^+, M^- \rangle$ be an Herbrand model of $SNF(\Gamma(C_i^{\top}))$. We define an S_i-3-interpretation $N = \langle N^+, N^- \rangle$ of $SNF(\Gamma(C))$ according to the following rules:

- for each atom $\alpha \in S_i$:
  - $\alpha \in M^+ \Longrightarrow \alpha \in N^+$;
  - $\alpha \in M^- \Longrightarrow \alpha \in N^-$;
- for each atom $\alpha \in H_b \setminus S_i$: $\alpha \in N^+$ and $\alpha \in N^-$.

Notice that N is necessarily an S_i-3-interpretation of $SNF(\Gamma(C))$, since the Herbrand Base of $SNF(\Gamma(C_i^{\top}))$ is equal to S_i. We now show that N is also an S_i-3-model, thus proving that C is S_i-3-satisfiable. Since $SNF(\Gamma(C))$ is an universally quantified formula, it is sufficient to show that N satisfies all its ground instances. We split the proof in two subcases:

1. all the variables of $SNF(\Gamma(C))$ are bound to terms belonging to U_i. In this case for every instance g of $SNF(\Gamma(C))$ there exists a corresponding instance h of $SNF(\Gamma(C_i^{\top}))$, where the same variables are bound to the same terms. More precisely, all the atoms of h belong to S_i, and h is obtained from g by substituting some of its subformulae with $\top$. All the atoms occurring in these subformulae are instantiated on terms of $U_k \setminus U_i$, where $k > i$ and U_k is a stratum of the Herbrand Base of $SNF(\Gamma(C))$. Therefore all these atoms belong to $S_k \setminus S_i$, hence are mapped into *inconsistent* by N. Since $M \models SNF(\Gamma(C_i^{\top}))$, we know that $M \models h$, hence $N \models h$. Since g differs from h in some literals which are anyway mapped into *inconsistent* by N, it follows that $N \models g$.
2. at least one variable of $SNF(\Gamma(C))$ is bound to a term not belonging to U_i. By definition of N, we know that all the atoms instantiated to those terms are mapped by N to *inconsistent*. Since the Herbrand Universe of $SNF(\Gamma(C_i^{\top}))$ is equal to U_i, some instances g of $SNF(\Gamma(C))$ do not have a corresponding instance of $SNF(\Gamma(C_i^{\top}))$. Let us consider an instance g' of $SNF(\Gamma(C_i^{\top}))$ obtained from g by binding any variable not bound to terms belonging to U_i to terms of U_i in an arbitrary way. From the previous item we know that $N \models g'$. Remember all the atoms instantiated to terms not belonging to U_i are mapped into *inconsistent* by N. Therefore g differs from g' in some literals which are mapped into *inconsistent* by N, hence it follows that $N \models g$.

(if part) Suppose that C is S_i-3-satisfiable, i.e. that $SNF(\Gamma(C))$ is S_i-3-satisfiable. Let $N = \langle N^+, N^- \rangle$ be an S_i-3 Herbrand model of $SNF(\Gamma(C))$. We define an Herbrand interpretation $M = \langle M^+, M^- \rangle$ of $SNF(\Gamma(C_i^{\top}))$ according to the following rule. For each atom $\alpha \in S_i$:

- $\alpha \in N^+ \Longrightarrow \alpha \in M^+$;

- $\alpha \in N^- \Longrightarrow \alpha \in M^-$;

Notice that M is necessarily an Herbrand interpretation of $SNF(\Gamma(C_i^\top))$, since the Herbrand Base of $SNF(\Gamma(C_i^\top))$ is equal to S_i. We now show that M is also an Herbrand model, thus proving that $C_i^\top$ is satisfiable. Since $SNF(\Gamma(C_i^\top))$ is an universally quantified formula, it is sufficient to show that M satisfies all its ground instances g. Let h be any instance of $SNF(\Gamma(C))$ which corresponds to g, where the same variables are bound to the same terms. Since $N \models SNF(\Gamma(C))$, we know that $N \models h$. Moreover h is obtained from g by substituting each occurrence of $\top$ with a formula. Since $\top$ occurs always positively in g and is satisfied by M, it follows that $M \models g$. □

Dealing with an S-1-interpretation M, when an atom $\alpha \in B_h \setminus S$ belongs neither to M^+ nor to M^-, we say that α is mapped into *undefined*.

PROOF of Theorem 4:
(only if part) Suppose that $C_i^\perp$ is satisfiable, i.e. that $SNF(\Gamma(C_i^\perp))$ is satisfiable. Let $M = \langle M^+, M^- \rangle$ be an Herbrand model of $SNF(\Gamma(C_i^\perp))$. We define an S_i-1-interpretation $N = \langle N^+, N^- \rangle$ of $SNF(\Gamma(C))$ according to the following rules:

- for each atom $\alpha \in S_i$:
  - $\alpha \in M^+ \Longrightarrow \alpha \in N^+$;
  - $\alpha \in M^- \Longrightarrow \alpha \in N^-$;
- for each atom $\alpha \in H_b \setminus S_i$: $\alpha \notin N^+$ and $\alpha \notin N^-$.

Notice that N is necessarily an S_i-1-interpretation of $SNF(\Gamma(C))$, since the Herbrand Base of $SNF(\Gamma(C_i^\perp))$ is equal to S_i. We now show that N is also an S_i-1-model, thus proving that C is S_i-1-satisfiable. Since $SNF(\Gamma(C))$ is an universally quantified formula, it is sufficient to show that N satisfies all its ground instances. We recall that, by definition of S-1-satisfiability, we have only to consider ground instances of $SNF(\Gamma(C))$ in which variables are substituted by terms of the set $Terms(S_i)$, which are the terms occurring in the set S_i, i.e. are the terms of U_i. Therefore we know that all the variables of $SNF(\Gamma(C))$ are bound to terms belonging to U_i. This implies that for every instance g of $SNF(\Gamma(C))$ there exists a corresponding instance h of $SNF(\Gamma(C_i^\perp))$, where the same variables are bound to the same terms. More precisely, all the atoms of h belong to S_i, and h is obtained from g by substituting some of its subformulae with $\perp$. Since $M \models SNF(\Gamma(C_i^\top))$, we know that $M \models h$, hence $N \models h$. Notice that $\perp$ is not satisfied by N, therefore $N \models g$ even if the subformulae in which g differs from h are not satisfied. Since this is the "worst" case, it follows that $N \models g$.

(if part) Suppose that C is S_i-1-satisfiable, i.e. that $SNF(\Gamma(C))$ is S_i-1-satisfiable. Let $N = \langle N^+, N^- \rangle$ be an S_i-1 Herbrand model of $SNF(\Gamma(C))$. We define an Herbrand interpretation $M = \langle M^+, M^- \rangle$ of $SNF(\Gamma(C_i^\perp))$ according to the following rule. For each atom $\alpha \in S_i$:

- $\alpha \in N^+ \Longrightarrow \alpha \in M^+$;
- $\alpha \in N^- \Longrightarrow \alpha \in M^-$;

Notice that M is necessarily an Herbrand interpretation of $SNF(\Gamma(C_i^\perp))$, since the Herbrand Base of $SNF(\Gamma(C_i^\perp))$ is equal to S_i. We now show that M is also an Herbrand model, thus showing that $C_i^\perp$ is satisfiable. Since $SNF(\Gamma(C_i^\perp))$ is an universally quantified formula, it is sufficient to show that M satisfies all its ground instances g. Let h be any instance of $SNF(\Gamma(C))$ which corresponds to g, where the same variables are bound to the same terms. Since $N \models SNF(\Gamma(C))$, we know that $N \models h$. Moreover h is obtained from g by substituting each occurrence of $\perp$ with a formula. All the atoms occurring in this formula are instantiated on terms of $U_k \setminus U_i$, where $k > i$ and U_k is a stratum of the Herbrand Base of $SNF(\Gamma(C))$. Therefore all these atoms belong to $S_k \setminus S_i$, hence are mapped into *undefined* by N. Since M maps $\perp$ into 0, it follows that $M \models g$. □

PROOF of Theorem 5:
The same proof of Theorem 1 applies also in this case. In fact, we only allow negation in front of primitive concepts. Hence, we never replace subconcepts that are in the scope of the $\neg$ operator.

PROOF of Theorem 7:
The proof is very similar to that of Theorem 3 with the additional complication of replacing some of the primitive concepts and their negation with $\top$. This is however already taken into account when we define the Herbrand base H_b of a simplified concept $C_{P,i}^\top$. In fact, H_b will only contain atoms of concepts appearing in $C_{P,i}^\top$ and, even in this case, is equal to $S_{P,i}$. Hence, the same proof of Theorem 3 applies.

PROOF of Theorem 8:
The proof is very similar to that of Theorem 4 since similar considerations to the ones done in the proof of Theorem 7 hold.

Adding Epistemic Operators to Concept Languages

Francesco M. Donini, Maurizio Lenzerini
Daniele Nardi, Andrea Schaerf
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy

Werner Nutt
German Research Center for
Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3
D-6600 Saarbrücken, Germany

Abstract

We investigate the use of epistemic operators in the framework of concept languages (also called terminological languages). The results of this work have a twofold significance. From the point of view of epistemic logics, our contribution is to have identified an effective procedure for the problem of answering epistemic queries posed to a knowledge base expressed in the concept language $\mathcal{ALC}$. From the point of view of concept languages, the most relevant aspect of our work is that we have reconstructed in logic several common features of existing knowledge representation systems. Epistemic operators provide a highly expressive query language; allow for the treatment of several database features, such as closed world reasoning and integrity constraints; and finally can give a formal characterization of some procedural mechanisms, such as trigger rules, usually considered in frame-based systems.

1 Introduction

A substantial amount of research has been carried out in the last decade with the aim of providing a logical reconstruction of frame-based knowledge representation languages. Much of this work has taken place in the context of *concept languages* (also called terminological languages or term subsumption languages). An important part of it has been concerned with both the study of reasoning techniques (e.g. subsumption algorithms) in concept languages, and the characterization of their computational complexity. Recent results [3, 4, 5, 13, 18] indicate that the crucial properties of a first order characterization of concept languages are now well understood, with regard to both expressive power of language constructs, and computational complexity of reasoning.

Nevertheless, it is a common opinion that several issues have still to be addressed in order to build principled practical systems based on concept languages (see for example [7, 20]). The following issues, among others, are considered especially important:

- providing knowledge representation systems based on concept languages with powerful and sophisticated querying capabilities;
- making concept languages able to interface with databases;
- providing a formal account of several mechanisms generally used in implemented systems, such as trigger rules [2, 21] or default rules, that are not directly expressible in first order logic.

Recent work on data and knowledge bases exploits the use of epistemic operators for improving both the expressiveness of knowledge representation languages and their associated querying facilities. In particular, it has been argued that queries should be permitted to address aspects of the external world as represented by the knowledge base, as well as aspects of what the knowledge base knows about the external world (see [10, 12, 16]).

The need for such a distinction is evident when a knowledge base contains incomplete information about individuals. Incompleteness may come in via existential quantification: for example, a knowledge base may know that the individual **susan** has at least one male friend (because it was told so), without knowing who this friend is. Incompleteness may also arise from disjunction: for example, a knowledge base may know that **andrea** is a person, and that all persons are either male or female,[1] without knowing exactly **andrea**'s sex.

The aim of our work is to propose a formal framework for adding epistemic operators to concept languages. Our opinion is that such a framework is the appropriate one for studying several extensions of concept lan-

[1] Notice that this is just a piece of knowledge and not an integrity constraint.

guages in order to go beyond their first order formalization, and to meet the most important requirements for the design of principled practical systems based on concept languages. Here we focus on the use of epistemic operators both in the query language, which allows also for the treatment of integrity constraints and closed-world reasoning, and in the formalization of trigger rules.

The basic building block of our proposal is the definition (Section 2) of a new concept language, called $\mathcal{ALCK}$, obtained from $\mathcal{ALC}$ [18] (a powerful concept language including conjunction, disjunction, and negation of concepts, together with existential and universal quantification on roles) by adding an epistemic operator in the spirit of [10, 12, 16]. One result of our work is the design of a technique (Section 3) for answering epistemic queries expressed in $\mathcal{ALCK}$. The technique is an extension of the tableaux-based method described in [4], which has been proved extremely useful for solving several reasoning and complexity problems in concept languages.

Our study shows that epistemic operators can be very useful for the design of more powerful knowledge representation systems based on concept languages. In particular, we show that epistemic operators enhance the expressive power of query languages, without increasing the computational complexity of query answering (Section 4). In addition we have found interesting cases where the use of epistemic operators allows one to express queries (not expressible in first order logic) that both have natural interpretations and are strictly less costly than their first order counterparts. Moreover, epistemic operators make several basic features of database systems available within concept languages (Section 5), allowing for the construction of knowledge representation systems where part of the knowledge is expressed using database models, with no mismatch between different portions of the knowledge base. Specifically, such operators can be used to query the knowledge base under the closed world assumption, to express integrity constraints over a knowledge base, and to make concept languages at least as expressive as relational database languages. We also show (Section 6) how $\mathcal{ALCK}$ can be used to provide a semantical characterization of a procedural mechanism considered in several frame-based systems, namely, trigger rules.

2 Epistemic Concepts and Knowledge Bases

We make use of the concept language $\mathcal{ALC}$ (see [4, 18]) to define a knowledge base.[2] Concept languages allow one to express the knowledge about the classes of interest in a particular application, through the notions of *concept* and *role*. Intuitively, concepts represent the classes of objects in the domain of interest, while roles represent relationships between concepts. Complex concept expressions can be defined by means of a number of language operators, applied to primitive concepts and roles.

The syntax and semantics of $\mathcal{ALC}$ are as follows. We assume that two alphabets of symbols, one for *primitive concepts*, and one for *primitive roles*, are given. The letter A will always denote a primitive concept, and the letter P will denote a role, which in $\mathcal{ALC}$ is always primitive. The *concepts* (denoted by the letters C and D) of the language $\mathcal{ALC}$ are built out of primitive concepts and primitive roles according to the syntax rule:

$$
\begin{array}{lll}
C, D \longrightarrow & A \mid & \text{(primitive concept)} \\
& \top \mid & \text{(top)} \\
& \bot \mid & \text{(bottom)} \\
& C \sqcap D \mid & \text{(intersection)} \\
& C \sqcup D \mid & \text{(union)} \\
& \neg C \mid & \text{(complement)} \\
& \forall P.C \mid & \text{(universal quantification)} \\
& \exists P.C & \text{(existential quantification).}
\end{array}
$$

In the following, we use parentheses whenever we need to disambiguate concept expressions. For example, we shall write $(\exists P.D) \sqcap E$ to mean that the concept E is not in the scope of $\exists P$.

Let Δ, called the *domain*, be a countably infinite set of symbols $p_1, p_2, \ldots$, called *parameters*. An *interpretation* $\mathcal{I}$ is a function that maps every concept to a subset of Δ and every role to a subset of $\Delta \times \Delta$, in such a way that the following equations are satisfied:

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta \\
\bot^{\mathcal{I}} &= \emptyset \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta \setminus C^{\mathcal{I}} \\
(\forall P.C)^{\mathcal{I}} &= \{p_1 \in \Delta \mid \\
&\qquad \forall p_2 : (p_1, p_2) \in P^{\mathcal{I}} \rightarrow p_2 \in C^{\mathcal{I}}\} \\
(\exists P.C)^{\mathcal{I}} &= \{p_1 \in \Delta \mid \\
&\qquad \exists p_2 : (p_1, p_2) \in P^{\mathcal{I}} \wedge p_2 \in C^{\mathcal{I}}\}.
\end{aligned}
$$

An interpretation $\mathcal{I}$ is a *model* for a concept C if $C^{\mathcal{I}}$ is nonempty. A concept is *satisfiable* if it has a model and *unsatisfiable* otherwise. We say that C is *subsumed* by D if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every interpretation $\mathcal{I}$.

Let $\mathcal{O}$ be an alphabet of symbols, called *individuals*. Syntactically, instance-of relationships are expressed in terms of *membership assertions*, each one either of the form $C(a)$ (meaning a is an instance of C) or $P(a, b)$ (meaning a is related to b by means of P),

[2] Although we restrict our attention to $\mathcal{ALC}$, our framework can be applied to other languages as well.

where a and b are individuals, C is a concept, and P is a role.

Formally, the meaning of membership assertions is specified as follows. Let γ be a *fixed*, injective function from $\mathcal{O}$ to Δ (i.e. each individual is associated with a unique parameter), and $\mathcal{I}$ be an interpretation. An assertion $C(a)$ is satisfied by $\mathcal{I}$ if $\gamma(a) \in C^{\mathcal{I}}$. Similarly, an assertion $P(a,b)$ is satisfied by $\mathcal{I}$ if $(\gamma(a),\gamma(b)) \in P^{\mathcal{I}}$. A finite set of membership assertions is called an $\mathcal{ALC}$-knowledge base. An interpretation $\mathcal{I}$ is a *model* of a knowledge base Σ if $\mathcal{I}$ satisfies all its assertions. Σ is satisfiable if it has a model. The set of models of Σ is denoted as $\mathcal{M}(\Sigma)$. Σ logically implies σ (written $\Sigma \models \sigma$), where σ is a membership assertion, if every model in $\mathcal{M}(\Sigma)$ satisfies σ.

In so-called terminological systems, the knowledge base also includes an intensional part, called terminology, expressed in terms of concept definitions, which are usually assumed to be acyclic (i.e. in the definition of concept C neither direct nor indirect reference to C itself may occur). It is well known that any reasoning process over knowledge bases comprising an acyclic terminology can be reduced to a reasoning process over a knowledge base with an empty terminology, since one can substitute every concept name in the assertions with the corresponding definition (see [14] for a discussion on this topic). For this reason, without loss of generality we conceive a knowledge base as just a set of assertions.

In the standard approach, querying a knowledge base Σ means asking whether $C(a)$ (or $P(a,b)$) is logically implied by Σ. The semantics associated with concept languages is an open world semantics: the answer to a query Q will be `YES` if Q is true in every model of Σ, `NO` if Q is false in every model, and `UNKNOWN` otherwise.

The use of epistemic operators in the query language allows for a more sophisticated interaction with the system, as we will see in the next sections. Generally speaking, we follow [16], and use $\mathbf{K}\beta$ to mean that the system *knows* that β is true in every model of Σ. We now introduce the language $\mathcal{ALCK}$, which is an extension of $\mathcal{ALC}$ with epistemic operators and is defined by the following syntax (where C, D denote concepts, R denotes a role, A denotes a primitive concept and P a primitive role):

$$\begin{aligned} C, D &\longrightarrow A \mid C \sqcap D \mid C \sqcup D \mid \neg C \mid \\ &\qquad \forall R.C \mid \exists R.C \mid \mathbf{K}C \\ R &\longrightarrow P \mid \mathbf{K}P. \end{aligned}$$

We give the semantics of $\mathcal{ALCK}$ by adapting the semantics presented in [10, 12, 16] to the framework of concept languages. An *epistemic* interpretation is a pair $(\mathcal{I},\mathcal{W})$ where $\mathcal{I}$ is an interpretation and $\mathcal{W}$ is a set of interpretations such that the following equations are satisfied:[3]

$$\begin{aligned} \top^{\mathcal{I},\mathcal{W}} &= \Delta \\ \bot^{\mathcal{I},\mathcal{W}} &= \emptyset \\ A^{\mathcal{I},\mathcal{W}} &= A^{\mathcal{I}} \\ P^{\mathcal{I},\mathcal{W}} &= P^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I},\mathcal{W}} &= C^{\mathcal{I},\mathcal{W}} \cap D^{\mathcal{I},\mathcal{W}} \\ (C \sqcup D)^{\mathcal{I},\mathcal{W}} &= C^{\mathcal{I},\mathcal{W}} \cup D^{\mathcal{I},\mathcal{W}} \\ (\neg C)^{\mathcal{I},\mathcal{W}} &= \Delta \setminus C^{\mathcal{I},\mathcal{W}} \\ (\forall R.C)^{\mathcal{I},\mathcal{W}} &= \{p_1 \in \Delta \mid \\ &\qquad \forall p_2.\,(p_1,p_2) \in R^{\mathcal{I},\mathcal{W}} \rightarrow p_2 \in C^{\mathcal{I},\mathcal{W}}\} \\ (\exists R.C)^{\mathcal{I},\mathcal{W}} &= \{p_1 \in \Delta \mid \\ &\qquad \exists p_2.\,(p_1,p_2) \in R^{\mathcal{I},\mathcal{W}} \wedge\ p_2 \in C^{\mathcal{I},\mathcal{W}}\} \\ (\mathbf{K}C)^{\mathcal{I},\mathcal{W}} &= \bigcap_{\mathcal{J} \in \mathcal{W}} (C^{\mathcal{J},\mathcal{W}}) \\ (\mathbf{K}P)^{\mathcal{I},\mathcal{W}} &= \bigcap_{\mathcal{J} \in \mathcal{W}} (P^{\mathcal{J},\mathcal{W}}). \end{aligned}$$

Notice that, since the domain is fixed independently of the interpretation, it is meaningful to refer to the intersection of the extensions of a concept in different interpretations. It follows that $\mathbf{K}C$ is interpreted in $\mathcal{W}$ as the set of objects that are instances of C in every model belonging to $\mathcal{W}$. In this sense, $\mathbf{K}C$ represents those objects *known* to be instances of C.

An $\mathcal{ALCK}$-knowledge base T is a finite set of membership assertions whose concepts and roles belong to the language $\mathcal{ALCK}$. An *epistemic model* of T is a pair $(\mathcal{I},\mathcal{W})$, where $\mathcal{I} \in \mathcal{W}$ and $\mathcal{W}$ is any maximal set of interpretations such that for each $\mathcal{J} \in \mathcal{W}$, every assertion of T is true in $(\mathcal{J},\mathcal{W})$. T is said to be *satisfiable* if there exists an epistemic model of T, *unsatisfiable* otherwise. T logically implies an assertion σ if σ is true in every epistemic model of T. Note that, if Σ does not contain epistemic operators, it is an $\mathcal{ALC}$-knowledge base, and its epistemic models are the pairs $(\mathcal{I},\mathcal{W})$, such that $\mathcal{W}$ is the set $\mathcal{M}(\Sigma)$ of ordinary models of Σ.

Let T be an $\mathcal{ALCK}$-knowledge base and let Σ be an $\mathcal{ALC}$-knowledge base (hence Σ is first order). An epistemic Σ-model of T is a pair $(\mathcal{I},\mathcal{M}(\Sigma))$ with $\mathcal{I} \in \mathcal{M}(\Sigma)$ such that every assertion in T is true in $(\mathcal{I},\mathcal{M}(\Sigma))$. T is said to be *Σ-satisfiable* if there exists an epistemic Σ-model of T, *Σ-unsatisfiable* otherwise. Let C be an $\mathcal{ALCK}$-concept, we write $T \not\approx_{\Sigma} C(a)$ to mean that every epistemic Σ-model of T satisfies $C(a)$, and we write $\Sigma \not\approx C(a)$ to mean $\Sigma \not\approx_{\Sigma} C(a)$.[4] It is easy to verify that $\Sigma \not\approx_{\Sigma} C(a)$ if and only if the $\mathcal{ALCK}$-knowledge base $\Sigma \cup \{\neg C(a)\}$ is Σ-unsatisfiable. Let us now introduce the notion of answer to a query.

[3] Notice that if one discards $\mathbf{K}$ and $\mathcal{W}$ in the equations, one obtains the standard semantics of $\mathcal{ALC}$.

[4] Since Σ is first order, this means that for every model $\mathcal{I} \in \mathcal{M}(\Sigma)$ the membership $C(a)$ is true in $(\mathcal{I},\mathcal{M}(\Sigma))$.

Definition 2.1 *Given an $\mathcal{ALC}$-knowledge base Σ, an $\mathcal{ALCK}$-concept C, and an individual a, the answer to the query $C(a)$ posed to Σ is* YES *if $\Sigma \not\models C(a)$,* NO *if $\Sigma \not\models \neg C(a)$, and* UNKNOWN *otherwise. Moreover, the* answer set *of C w.r.t. Σ is the set $\{a \in \mathcal{O}_\Sigma \mid \Sigma \not\models C(a)\}$, where $\mathcal{O}_\Sigma$ is the set of individuals appearing in Σ.*

We end this section with an example illustrating three epistemic queries.

Example 2.2 Consider the $\mathcal{ALC}$-knowledge base Σ_1 and the following three $\mathcal{ALCK}$-queries:

$\Sigma_1 = \{$FRIEND(john, susan), FRIEND(john, peter),
LOVES(susan, peter), LOVES(peter, mary)
$\forall$LOVES.$\neg$Male(john),
Married $\sqcap$ $\exists$FRIEND.Male(susan),
Male(peter), $\neg$Married(mary)$\}$.

Query 1: $\exists$FRIEND.Male(susan), i.e., is there a friend of susan's who is male? Answer: YES.

Query 2: $\exists$KFRIEND.KMale(susan), i.e., is there someone that is known to be both a friend of susan's and male? Answer: NO.

Query 3: $\exists$KLOVES.$\neg$KMarried(susan), i.e., is there someone that is known to be loved by susan that is not known to be married? Answer: YES (peter).

The answer to Query 1 is justified by the explicit assertion $\exists$FRIEND.Male(susan). The answer to Query 2 is explained by the fact that $\{p \mid (\gamma(\text{susan}), p) \in (\text{KFRIEND})^{\mathcal{I},\mathcal{W}}\}$ is empty, for any model $(\mathcal{I}, \mathcal{W})$ of Σ_1. In other words, the knowledge base knows that susan has a male friend, but it does not know who he is, and therefore, when taking the intersection of the extensions of FRIEND in the models of Σ_1, all the pairs $(\gamma(\text{susan}), p)$ are ruled out. On the other hand, the answer to Query 3 is YES, since the knowledge base knows that susan loves peter, and it does not know that peter is married, because peter is not in $(\text{KMarried})^{\mathcal{I},\mathcal{W}}$.

3 The Calculus for Query Answering

The problem of designing methods for epistemic query answering is dealt with in [10, 16]. In [16], a procedure is presented that is sound and complete if the query satisfies some syntactic constraints. However, not all epistemic concepts belonging to $\mathcal{ALCK}$ satisfy such constraints (for example, the formula corresponding to $\exists P.\neg \mathbf{K} C(a)$ is not admissible in [16]). On the other hand, the method proposed in [10] has been conceived within a more general framework, and its specialization to the case of concept languages does not provide an effective procedure. It follows that none of these approaches can be directly applied in our setting.

As we said in Section 2, one way to answer epistemic queries posed to an $\mathcal{ALC}$-knowledge base Σ is to check whether the $\mathcal{ALCK}$-knowledge base obtained by adding to Σ the negation of the query is Σ-unsatisfiable. In this section we present a general method for the problem of checking whether an $\mathcal{ALCK}$-knowledge base is Σ-satisfiable, where Σ is an $\mathcal{ALC}$-knowledge base.

Let $\mathcal{V}$ be a set of *variables*. The elements of $\mathcal{V}$ will be denoted by x, y, while the elements of $\mathcal{O} \cup \mathcal{V}$ (called *objects*) will be denoted by w, z. Finally, the elements of $\mathcal{O}$ will be denoted by a, b. A *constraint* is a syntactic structure of one of the forms:

$$w{:}C, \quad wRz,$$

where C is an $\mathcal{ALCK}$-concept and R is an $\mathcal{ALCK}$-role. A *constraint system* is a finite set of constraints of the above forms.

In order to assign a meaning to constraints, we need the following definitions. An *assignment* $\alpha(\cdot)$ is a function from $\mathcal{O} \cup \mathcal{V}$ into Δ, such that $\alpha_{|\mathcal{O}} = \gamma$, i.e. α is an extension of γ to variables. Let $(\mathcal{I}, \mathcal{W})$ be an epistemic interpretation, and let α be an assignment. The triple $(\mathcal{I}, \mathcal{W}, \alpha)$ is said to satisfy the constraint $w{:}C$ if $\alpha(w) \in C^{\mathcal{I},\mathcal{W}}$. Similarly, $(\mathcal{I}, \mathcal{W}, \alpha)$ satisfies the constraint wRz if $(\alpha(w), \alpha(z)) \in R^{\mathcal{I},\mathcal{W}}$. Let S be a constraint system, and let Σ be an $\mathcal{ALC}$-knowledge base. $(\mathcal{I}, \mathcal{W}, \alpha)$ is a *solution* of S if $(\mathcal{I}, \mathcal{W}, \alpha)$ satisfies all of its constraints. S is said to be Σ-*solvable* if there is a triple $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ that is a solution of S. If there is no such solution, then S is said to be Σ-*unsolvable*.

Given an $\mathcal{ALC}$-knowledge base Σ, we define S_Σ to be the constraint system that includes one constraint $a{:}C$ for each assertion $C(a)$ of Σ, and one constraint aRb for each assertion $R(a, b)$ of Σ (see [8]). The next proposition shows that answering a query posed to a knowledge base Σ can be reduced to checking a particular constraint system for Σ-solvability.

Proposition 3.1 *Let Σ be an $\mathcal{ALC}$-knowledge base, C be an $\mathcal{ALCK}$-concept, and a an individual. Then $\Sigma \not\models C(a)$ iff $S_\Sigma \cup \{a{:}\neg C\}$ is Σ-unsolvable.*

An $\mathcal{ALCK}$-concept is said to be simple if every complement appearing in it is either of the form $\neg A$ or of the form $\neg \mathbf{K} C$, where C is simple. Every $\mathcal{ALCK}$-concept can be rewritten in linear time into an equivalent simple concept. In the rest of the paper we assume to deal only with simple concepts.

We say that wRz *holds* in a constraint system S if either

1. R is P, and $wPz \in S$, or
2. R is $\mathbf{K}P$, $w, z \in \mathcal{O}$, and $wPz \in S$.

When checking the satisfiability of a constraint system, the following set of *completion rules* are used (S denotes a constraint system):

1. $S \rightarrow_{\sqcap} \{w:C_1,\ w:C_2\} \cup S$
   if $w:C_1 \sqcap C_2$ is in S, and $w:C_1$ and $w:C_2$ are not both in S;
2. $S \rightarrow_{\sqcup} \{w:D\} \cup S$
   if $w:C_1 \sqcup C_2$ is in S, neither $w:C_1$ nor $w:C_2$ is in S, and $D = C_1$ or $D = C_2$;
3. $S \rightarrow_{\exists} \{wRx,\ x:C\} \cup S$
   if $w:\exists R.C$ is in S, there is no z such that both wRz and $z:C$ are in S and x is a new variable;
4. $S \rightarrow_{\forall} \{z:C\} \cup S$
   if $w:\forall R.C$ is in S, wRz holds in S, and $z:C$ is not in S.

Proposition 3.2 *Let Σ be an $\mathcal{ALC}$-knowledge base, and S, S' be two constraint systems. Then:*

1. *If S' is obtained from S by application of one of rules $\rightarrow_{\sqcap}$, $\rightarrow_{\exists}$, $\rightarrow_{\forall}$, then S is Σ-solvable if and only if S' is Σ-solvable.*
2. *If S' is obtained from S by application of the $\rightarrow_{\sqcup}$-rule, then S is Σ-solvable if S' is Σ-solvable. Conversely, if S is Σ-solvable and the $\rightarrow_{\sqcup}$-rule is applicable to S, then it can be applied in such a way that it yields a Σ-solvable constraint system.*

A constraint system is said to be *complete* if no rule is applicable to it. Any complete constraint system obtained from a constraint system S by applying the above rules is called a *completion* of S. Notice that, due to the presence of the $\rightarrow_{\sqcup}$-rule, more than one completion can be obtained starting from a constraint system.

We denote by $\mathcal{O}_S$ the set of individuals appearing in a constraint system S. Moreover, if a is an individual, then we denote by $S[x/a]$ the constraint system obtained from S by substituting every occurrence of the variable x with the individual a.

We now introduce the notion of Σ-clash, which intuitively corresponds to an explicit contradiction in constraint systems. Then, we exploit such a notion in order to derive a sufficient and necessary condition for a constraint system to be Σ-solvable.

Definition 3.3 *A constraint system S contains a Σ-clash if at least one of the following conditions holds:*

1. *S contains a constraint of the form $w:\bot$;*
2. *S contains two constraints of the form $w:A, w:\neg A$;*
3. *S contains a constraint of the form $a:\mathbf{K}C$, and there is at least one completion of $S_\Sigma \cup \{a:\neg C\}$ without Σ-clashes;*
4. *S contains a constraint of the form $a:\neg\mathbf{K}C$, and every completion of $S_\Sigma \cup \{a:\neg C\}$ contains a Σ-clash;*
5. *S contains a constraint of the form $a\mathbf{K}Pb$, and $aPb \notin S_\Sigma$;*
6. *S contains $x:\neg\mathbf{K}C$, $x:\mathbf{K}C$, $x\mathbf{K}Py$, or $w\mathbf{K}Px$, and for each $a \in \mathcal{O}_S \cup \{\iota\}$, every completion of $S[x/a]$ contains a Σ-clash, where ι is an individual in $\mathcal{O} \setminus \mathcal{O}_S$.*

Proposition 3.4 *Let Σ be an $\mathcal{ALC}$-knowledge base, and S be a constraint system. Then S is Σ-solvable if and only if there exists at least one completion of S that contains no Σ-clash.*

Proof. The proof is by induction on the number k of occurrences of the epistemic operator in the constraint system. If $k = 0$, then the claim trivially follows from the results in [1]. If $k > 0$, then let H_1 be the following induction hypothesis: any complete constraint system with $h < k$ occurrences of the epistemic operator is Σ-solvable if and only if it contains no Σ-clash. Let S be a complete constraint system with k occurrences of the epistemic operator. We must show that

1. if S contains no Σ-clash, then it is Σ-solvable, and
2. if S is Σ-solvable, then it contains no Σ-clash.

Proof of (1). Suppose S contains no Σ-clash. We construct a triple $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$. First of all, α is defined by:

- $\alpha(w) = \gamma(w)$ if w is an individual;
- $\alpha(x) = \gamma(a)$ if w if x is involved in at least one epistemic constraint in S (i.e., such that $x:\neg\mathbf{K}C$, $x:\mathbf{K}C$, $x\mathbf{K}Py$, or $w\mathbf{K}Px$ is in S), and a is any individual in $\mathcal{O}_S \cup \{\iota\}$ such that $S[x/a]$ does not contain any Σ-clash (there must be such an a, otherwise S would contain a clash of type 6);
- $\alpha(x) = p$ if x is not involved in any epistemic constraint, and p is any parameter that is assigned by α neither to an individual in $\mathcal{O}_S$ nor to a variable y in S such that $y \neq x$.

Secondly, $\mathcal{I}$ is defined as follows:

- for every primitive concept A and every parameter p, let $p \in A^{\mathcal{I}}$ if and only if there is a w such that $\alpha(w) = p$, and $w:A$ is in S;
- for every primitive role P and every pair p_1, p_2 of parameters, let $(p_1, p_2) \in P^{\mathcal{I}}$ if and only if there are w, z such that $\alpha(w) = p_1$, $\alpha(z) = p_2$, and wPz is in S.
- the interpretations of complex concepts and roles are derived from the semantic equations given in Section 2.

We show that $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ satisfies every constraint in S, i.e. S is Σ-solvable.

Consider any constraint of the form wPz. By the construction of α and $\mathcal{I}$, it is easy to verify that $(\alpha(w), \alpha(z)) \in P^{\mathcal{I},\mathcal{M}(\Sigma)}$.

Consider any constraint of the form $w{:}C$. We proceed by a secondary induction on the form of C, considering as base cases $\top$, A, $\neg A$, $\mathbf{K}C$, and $\neg\mathbf{K}C$.

If C is of one of the forms $\top$, A, $\neg A$, then it follows by the construction of α and $\mathcal{I}$ that $\alpha(w) \in C^{\mathcal{I},\mathcal{M}(\Sigma)}$.

Consider any constraint of the form $a{:}\neg\mathbf{K}C$. Since S does not contain any Σ-clash, there is at least one completion of $S_\Sigma \cup \{a{:}\neg C\}$ which does not contain any Σ-clash. Since the number of occurrences of the epistemic operator in $S_\Sigma \cup \{a{:}\neg C\}$ is less than k, by the induction hypothesis H_1 and by Proposition 3.2, $S_\Sigma \cup \{a{:}\neg C\}$ is Σ-solvable, which means that there is a model $\mathcal{J}$ of Σ, such that $\gamma(a) \in (\neg C)^{\mathcal{J},\mathcal{M}(\Sigma)}$, and hence $\gamma(a) \notin C^{\mathcal{J},\mathcal{M}(\Sigma)}$. Since $\mathcal{J} \in \mathcal{M}(\Sigma)$, $\gamma(a) \notin \bigcap_{\mathcal{J}\in\mathcal{M}(\Sigma)} C^{\mathcal{J},\mathcal{M}(\Sigma)}$, hence, by definition of $(\mathbf{K}C)^{\mathcal{I},\mathcal{M}(\Sigma)}$, $\gamma(a) \notin (\mathbf{K}C)^{\mathcal{I},\mathcal{M}(\Sigma)}$. Therefore, $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ satisfies $a{:}\neg\mathbf{K}C$.

Consider any constraint of the form $a{:}\mathbf{K}C$. Since S does not contain any Σ-clash, every completion of $S_\Sigma \cup \{a{:}\neg C\}$ contains a Σ-clash. Since the number of occurrences of the epistemic operator in $S_\Sigma \cup \{a{:}\neg C\}$ is less than k, by the induction hypothesis H_1 and by Proposition 3.2, $S_\Sigma \cup \{a{:}\neg C\}$ is unsolvable, which means that for every model $\mathcal{J}$ of Σ, $\gamma(a) \in C^{\mathcal{J},\mathcal{M}(\Sigma)}$, i.e. $\gamma(a) \in \bigcap_{\mathcal{J}\in\mathcal{M}(\Sigma)} C^{\mathcal{J},\mathcal{M}(\Sigma)}$, and hence $\gamma(a) \in (\mathbf{K}C)^{\mathcal{I},\mathcal{M}(\Sigma)}$. Therefore, $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ satisfies $a{:}\mathbf{K}C$.

Consider any constraint of the form $x{:}\mathbf{K}C$. Let a be the individual in $\mathcal{O}_S \cup \{\iota\}$ such that $\alpha(x) = \gamma(a)$ (notice that by the construction of α, $S[x/a]$ does not contain any Σ-clash). Since $S[x/a]$ does not contain any Σ-clash, every completion of $S_\Sigma \cup \{a{:}\neg C\}$ contains a clash. Recall from the previous case that the last condition implies $\gamma(a) \in \bigcap_{\mathcal{J}\in\mathcal{M}(\Sigma)} C^{\mathcal{J},\mathcal{M}(\Sigma)}$, and hence $\gamma(a) \in (\mathbf{K}C)^{\mathcal{I},\mathcal{M}(\Sigma)}$. It follows that $\alpha(x) \in (\mathbf{K}C)^{\mathcal{I},\mathcal{M}(\Sigma)}$, that is, $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ satisfies $x{:}\mathbf{K}C$. The other forms of epistemic constraints, namely $x{:}\neg\mathbf{K}C$, $x\mathbf{K}Py$, and $w\mathbf{K}Px$ can be treated analogously.

Now consider the following induction hypotesis H_2: for every proper subconcept D of a concept C, every constraint $z{:}D$ is satisfied by $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$.

Suppose $w{:}C$ is in S and C has the form $D \sqcap E$. Since S is complete, both $w{:}D$ and $w{:}E$ are in S. By the induction hypothesis H_2, $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ satisfies both constraints, and therefore, $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ satisfies $w{:}D \sqcap E$ too. The remaining forms of constraints, namely, $E \sqcup D$, $\exists R.D$, and $\forall R.D$, can be treated analogously.

In conclusion, we have shown that the triple $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ is a solution of S, and therefore S is Σ-solvable.

Proof of (2). Suppose S contains a Σ-clash. We now consider each type of clash in turn, and show that if S contains a clash of that type, then it is Σ-unsolvable.

1, 2. If S contains a clash of type 1 or 2, then it is clearly Σ-unsolvable.

3. Suppose S contains a constraint of the form $a{:}\mathbf{K}C$, and there is at least one completion of $S_\Sigma \cup \{a{:}\neg C\}$ without Σ-clash. By the induction hypothesis H_1 and by Proposition 3.2, $S_\Sigma \cup \{a{:}\neg C\}$ is Σ-solvable, i.e. there is a triple $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ that satisfies all the constraints of $S_\Sigma \cup \{a{:}\neg C\}$, and in particular $a{:}\neg C$. Therefore, $\gamma(a) \notin C^{\mathcal{I},\mathcal{M}(\Sigma)}$, which implies that $\gamma(a) \notin \bigcap_{\mathcal{J}\in\mathcal{M}(\Sigma)} C^{\mathcal{J},\mathcal{M}(\Sigma)}$. It follows that the constraint $a{:}\mathbf{K}C$ cannot be satisfied by any triple $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$, and therefore S is Σ-unsolvable.

4, 5. If S contains a Σ-clash of the form $a{:}\neg\mathbf{K}C$ or of the form $a\mathbf{K}Pb$, then we can proceed analogously to case 3.

6.1. Suppose S contains a constraint of the form $x{:}\neg\mathbf{K}C$, and for each $a \in \mathcal{O}_S \cup \{\iota\}$, $S[x/a]$ contains a Σ-clash, where ι is an individual in $\mathcal{O}$ not appearing in S. This means that for each $a \in \mathcal{O}_S \cup \{\iota\}$, either every completion of $S_\Sigma \cup \{a{:}\neg C\}$ contains a Σ-clash, or every completion of $S[x/a] \setminus \{a{:}\neg\mathbf{K}C\}$ contains a Σ-clash. By the induction hypothesis H_1, this implies that for each $a \in \mathcal{O}_S \cup \{\iota\}$, either $S_\Sigma \cup \{a{:}\neg C\}$ is Σ-unsolvable, or $S[x/a] \setminus \{a{:}\mathbf{K}C\}$ is Σ-unsolvable. We show that S cannot be satisfied by any triple $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$.

Let us assume that the triple $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ is a solution of S, with $\alpha(x) = p$, and there is a $b \in \mathcal{O}_S$ such that $\gamma(b) = p$. Obviously, $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ is a solution of $S[x/b]$ too, which means both that the constraint $b{:}\neg\mathbf{K}C$ is satisfied by $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$—i.e. that $S_\Sigma \cup \{b{:}\neg C\}$ is Σ-solvable—and that every constraint in $S[x/b] \setminus \{b{:}\neg\mathbf{K}C\}$ is satisfied by $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$—i.e. $S[x/b] \setminus \{b{:}\neg\mathbf{K}C\}$ is Σ-solvable—which is a contradiction.

Let us assume that the triple $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ is a solution of S, with $\alpha(x) = p$, and there is no $b \in \mathcal{O}_S$ such that $\gamma(b) = p$. It is possible to show that in this case there is a solution $(\mathcal{I}', \mathcal{M}(\Sigma), \alpha')$ of S such that $\alpha'(x) = \gamma(\iota)$. It follows that $(\mathcal{I}', \mathcal{M}(\Sigma), \alpha')$ is a solution of $S[x/\iota]$, too, which is again a contradiction as in the previous case.

6.2, 6.3, 6.4. If S contains a Σ-clash of the form $x{:}\mathbf{K}C$, $x\mathbf{K}Py$, or $w\mathbf{K}Px$, then we can proceed analogously to case 6.1. □

The results reported in [1] show that it is decidable whether a constraint system that does not include any epistemic constraint is Σ-solvable. In particular, it

is shown that the number of completions obtainable from such a system is finite. Based on this fact, one can easily prove that the number of completions of an $\mathcal{ALCK}$-constraint system is also finite. Observe that, in order to decide whether a complete constraint system S has a Σ-clash or not, a finite number of Σ-satisfiability checks suffices, each involving a constraint system whose number of epistemic constraints is less than the number in S. Therefore, one can show by induction that the above rules provide us with an algorithm for checking an $\mathcal{ALCK}$-constraint system for Σ-solvability.

Proposition 3.5 *Let Σ be an $\mathcal{ALC}$-knowledge base. Then it is decidable whether an $\mathcal{ALCK}$ constraint system S is Σ-solvable or not.*

This in turn implies that we have an effective method both for checking whether $\Sigma \models\!\!\!\sim C(a)$, and for computing the answer set of C w.r.t. Σ. The next section discusses in more detail the computational complexity of the method.

4 Complexity and Expressiveness

In this section we first address the computational complexity of answering $\mathcal{ALCK}$-queries. Then we discuss how the epistemic operator can be used in the formulation of interesting queries to an $\mathcal{ALC}$-knowledge base.

Using a technique based on constraint systems, both satisfiability and subsumption of $\mathcal{ALC}$-concepts are are proved in [18] to be PSPACE-complete problems. That paper also presents a linear space algorithm for these problems. The basic idea behind the algorithm is that, although the whole constraint system involved in the computation may have exponential size, it is possible to keep track of only a polynomial part of it at a time. These parts, called *traces*, are mutually independent, and can be checked for a clash separately.

In [1] a PSPACE algorithm for checking the satisfiability of an $\mathcal{ALC}$-knowledge base Σ is given, which again is based on the constraint system calculus. Since for any $\mathcal{ALC}$-concept C we have $\Sigma \models C(a)$ if and only if the $\mathcal{ALC}$-knowledge base $\Sigma \cup \{\neg C(a)\}$ is unsatisfiable, it follows that answering $\mathcal{ALC}$-queries to an $\mathcal{ALC}$-knowledge base can be done in polynomial space, too.

The algorithm for deciding the satisfiability of $\mathcal{ALC}$ knowledge bases consists of two steps: in the first one, which we call *precompletion*, all the information regarding each individual is collected in a single constraint; in the second step these constraints are separately tested for satisfiability using the linear-space algorithm for concept satisfiability given in [18].

However, when the query is an $\mathcal{ALCK}$-concept, the above method is no longer valid. In fact, because of the presence of the K operator, the satisfiability of constraint systems obtained from the precompletion cannot be done by considering the constraints on each individual separately. For example, the satisfiability of a constraint of the form $a{:}\exists \mathbf{K}P.\mathbf{K}C$ depends on the assertions on other individuals in the knowledge base. For instance, its satisfiability follows from the two assertions: $P(a,b)$ and $C(b)$.

In the calculus shown in Section 3, this possibility is taken into account by the condition for a Σ-clash of type 6. In fact, getting back to the above example, the constraint $a{:}\exists \mathbf{K}P.\mathbf{K}C$ is processed creating the two constraints $a\mathbf{K}Px$ and $x{:}\mathbf{K}C$, and checking whether x can be substituted with an individual without running into a clash.

Nevertheless, answering $\mathcal{ALCK}$-queries can be done in PSPACE. In fact, it is still possible to devise an algorithm that uses polynomial space by keeping a table with the information concerning substitutions. Such a table has a boolean value for every subconcept of Σ and of the query C, and every individual in $\mathcal{O}_\Sigma$. The value of an entry (D,a) is true whenever the completion process of $S_\Sigma \cup \{a{:}\neg C\}$ would add the constraint $a{:}D$ as the result of a substitution $[x/a]$ in a constraint $x{:}D$. Notice that a table represents the set of constraints on individuals that are common to all traces of a completion.

The algorithm generates a table and then considers the constraint system $S_\Sigma \cup \{a{:}\neg C\} \cup T$, where T is the set of constraints represented by the table. At this point the algorithm performs the same two steps of the algorithm for $\mathcal{ALC}$, but for a more complex verification of clashes in the traces, where all the six types described in the previous section are considered. Clashes of type 1,2,5 are found by inspection. Clashes of type 3,4 require a recursive call to the algorithm. Finally in the check of clashes of type 6, substitutions which introduce constraints not already present in the table are disallowed. If there are no allowed substitutions leading to a trace with no Σ-clash, the table is discarded and the algorithm restarts with a new one. If no table leads to the construction of a completion with no Σ-clash, the constraint system $S_\Sigma \cup \{a{:}\neg C\}$ is not Σ-solvable.

The size of the table is polynomially bounded by $(|\Sigma| + |C|) \times |\mathcal{O}_\Sigma|$, where $|\Sigma|$, $|C|$ and $|\mathcal{O}_\Sigma|$ denote the size of Σ, C and $\mathcal{O}_\Sigma$, respectively. Since each recursive call requires a new table but reduces the number of K operators, the number of tables that are simultaneously needed is bounded by the nesting of the K operator. Therefore, we can conclude that adding the K operator to the query language does not change the complexity class of query answering.

Proposition 4.1 *Let Σ be an $\mathcal{ALC}$-knowledge base, C be an $\mathcal{ALCK}$ concept, and a be an individual. Then checking whether $\Sigma \models\!\!\!\sim C(a)$ is PSPACE-complete.*

The analysis of the query answering algorithm allows us to make an interesting comparison between reasoning with epistemic concepts and reasoning with concepts involving collections of individuals. A construct for collection of individuals, sometimes called "ONE-OF," has been considered in [2] and [9]. If $a_1, \ldots, a_n$ are individuals, then the concept $\{a_1, \ldots, a_n\}$ is interpreted as the set consisting of the elements denoted by $a_1, \ldots, a_n$. Now, the intuition is that a concept of the form $\mathbf{K}C$ can be considered equivalent to the concept $\{a_1, \ldots, a_n\}$, where $a_1, \ldots, a_n$ are exactly the individuals for which $\Sigma \models C(a_i)$ holds. Indeed, we have proved that if $\mathcal{L}$ is any sublanguage of $\mathcal{ALC}$, $\mathcal{LK}$ is the language obtained from $\mathcal{L}$ by adding the constructs $\mathbf{K}C$, $\neg\mathbf{K}C$, and $\mathbf{K}P$, and $\mathcal{LO}$ is the language obtained from $\mathcal{L}$ by adding the constructs $\{a_1, \ldots, a_n\}$ and $\neg\{a_1, \ldots, a_n\}$ for concepts, and $\{(a_1, b_1), \ldots, (a_n, b_n)\}$ for roles, then the two problems of querying an $\mathcal{L}$-knowledge base using $\mathcal{LK}$-concepts and using $\mathcal{LO}$-concepts are in the same complexity class. In particular, Proposition 4.1 implies that reasoning in $\mathcal{ALCO}$ is PSPACE-complete, which represents a new complexity result for concept languages.

The previous considerations allow us to conclude that the use of epistemic operators does not substantially increase the complexity in query answering. This is extremely interesting, especially in light of the greater expressive power gained (see also next section). In the rest of this section we show how the use of the epistemic operator allows us to express queries that have both natural interpretation and good computational properties. In particular, we have found that there are situations where the use of the epistemic operator helps lessening the computational complexity of query answering.

Consider the following three queries posed to the $\mathcal{ALC}$-knowledge base Σ_1 of Example 2.2:

Query 4:
$\exists$FRIEND.(Married $\sqcap$ $\exists$LOVES.$\neg$Married)(john).
Answer: YES.

Query 5:
$\exists$KFRIEND.K(Married $\sqcap$ $\exists$LOVES.$\neg$Married)(john).
Answer: NO.

Query 6:
$\exists$KFRIEND.K(Married $\sqcap$ $\exists$LOVES.$\neg$KMarried)(john).
Answer: YES.

Query 4 asks whether john has a married friend who loves an unmarried person. At first glance, since susan and peter are the only known friends of john, it seems that the answer to the query is to be found by checking whether either $\Sigma_1 \models$ Married $\sqcap$ $\exists$LOVES.$\neg$Married(susan) or $\Sigma_1 \models$ Married $\sqcap$ $\exists$LOVES.$\neg$Married(peter).

Since $\Sigma_1 \not\models$ Married(peter), we have in particular that $\Sigma_1 \not\models$ Married $\sqcap$ $\exists$LOVES.$\neg$Married(peter), and since $\Sigma_1 \not\models \exists$LOVES.$\neg$Married(susan), it follows that $\Sigma_1 \not\models$ Married $\sqcap$ $\exists$LOVES.$\neg$Married(susan).

Reasoning in this way would lead to the answer NO. On the contrary, the correct answer to the query is YES, and in order to find it, one needs to reason by case analysis. In fact, the query asks if in every model M of Σ_1 there is an individual, say z, such that FRIEND(john, z), Married(z), and $\exists$LOVES.$\neg$Married(z) are true in M. Obviously, in every model M of Σ_1, either Married(peter) or $\neg$Married(peter) is true. In the first case, it is easy to see that z is simply peter (and the unmarried person he loves is mary), while in the second case z is susan (and the unmarried person she loves is just peter). Therefore, such an individual z exists in every model of Σ_1, and the query gets the answer YES.

Note that, even if none of the individuals related to the individual john through the role FRIEND is in the condition requested by the query, it happens that the combination of the assertions on the individuals (susan and peter) in the knowledge base is such that in every model either one or the other is in that condition.

On the other hand, Query 5 asks whether there is an individual, say z, such that in *every* model $M \in \mathcal{M}(\Sigma_1)$, both FRIEND(john, z) and Married $\sqcap$ $\exists$LOVES.$\neg$Married(z) are true. It is easy to see that no such z exists in Σ_1, and therefore the answer is NO.

Query 6 is similar to Query 5, but the concept $\neg$Married is replaced with the concept $\neg$KMarried. In this case, since $\neg$KMarried(peter) holds in Σ_1, we have that the above mentioned individual z exists. In particular, it is susan, and therefore the answer is YES.

Queries 4,5 and 6 show how the K operator is able to modify the semantics of a query (note that, apart from the presence of K, the queries are identical).

Queries 4, 5, and 6 show how to use the K operator to modify the semantics of a query (note that they differ only in the occurrences of the K operator): Query 4 respects the standard first order semantics of concept languages, Query 5 uses an intuitionistic semantics which rules out the reasoning by case analysis, and finally, Query 6 makes use of a sort of *negation as failure*.

As another example of this fact consider the following pair of queries:

Query 7: ($\exists$LOVES.$\neg$Male) $\sqcup$ ($\neg\exists$LOVES.$\top$)(john).
Answer: YES.

Query 8: K($\exists$LOVES.$\neg$Male) $\sqcup$ K($\neg\exists$LOVES.$\top$)(john).
Answer: NO.

The difference between Query 7 and 8 is similar to the difference between Query 4 and 5. Query 7 asks whether in every model $M \in \mathcal{M}(\Sigma_1)$ the assertion

$(\exists$LOVES.$\neg$Male$) \sqcup (\neg\exists$LOVES.$\top)$(john) is true.

Again, reasoning by case analysis, one realizes that such a query gets the answer YES: indeed, due to the assertion $\forall$LOVES.$\neg$Male(john) in Σ_1, for every model M of Σ_1, either $\exists$LOVES.$\neg$Male(john) is true in M, or $\neg\exists$LOVES.$\top$(john) is true in M. On the other hand, Query 8 asks whether it is the case that $\exists$LOVES.$\neg$Male(john) is true in every model $M \in \mathcal{M}(\Sigma_1)$, or whether $\neg\exists$LOVES.$\top$(john) is true in every model $M \in \mathcal{M}(\Sigma_1)$. Therefore, the answer to Query 8 is NO.

The above examples (Queries 5 and 8) show that the use of K may allow us to express queries whose answer does not require reasoning by case analysis. In [6], we analyze in detail the situations where this kind of reasoning makes deductions in concept languages problematic from the computational point of view. For example, we prove that the problem of checking if $\Sigma \models C(a)$, when Σ is an $\mathcal{AL}$-knowledge base and C is an $\mathcal{ALE}$-concept, is coNP-hard w.r.t. the size of Σ.[5]

On the other hand, we have the following result:

Proposition 4.2 *Let Σ be an $\mathcal{AL}$-knowledge base, a be an individual, and C be an $\mathcal{ALEK}$-concept where the only qualified existential quantifications are of the form $\exists \mathbf{K}R.\mathbf{K}E$. Then checking whether $\Sigma \not\models C(a)$ can be done in polynomial time w.r.t. the size of Σ.*

Notice that if C is an $\mathcal{ALE}$-concept, and $\Phi(C)$ is the $\mathcal{ALEK}$-concept obtained from C by replacing every occurrence of the construct $\exists R.E$ with $\exists \mathbf{K}R.\mathbf{K}E$, then $\Sigma \not\models \Phi(C)(a)$ implies $\Sigma \models C(a)$. From this observation and the above theorem we can derive an interesting property: the algorithm for checking whether $\Sigma \not\models \Phi(C)(a)$ is a tractable, sound and incomplete procedure for checking whether $\Sigma \models C(a)$.

Following the above idea, we can use the K operator to build sound and incomplete query answering procedures for any concept language. Given a knowledge base Σ and a query C, both expressed in a language $\mathcal{L}$, the query can be replaced by a suitable concept $\Psi_{\mathcal{L}}(C)$ in the language $\mathcal{LK}$ such that $\Sigma \not\models \Psi_{\mathcal{L}}(C)(a)$ implies $\Sigma \models C(a)$ and deciding $\Sigma \not\models \Psi_{\mathcal{L}}(C)(a)$ can be done more efficiently w.r.t. the size of Σ. The advantage compared to other incomplete procedures is the precise semantical characterization of the source of incompleteness.

[5] $\mathcal{ALE}$ is the sublanguage of $\mathcal{ALC}$ that consists of all simple concepts which do not contain the union constructor "$\sqcup$", whereas $\mathcal{AL}$ consists of all $\mathcal{ALE}$ concepts that only contain existential quantifications of the form $\exists R.\top$.

5 Closed World Reasoning, Relational Databases, and Integrity Constraints

The reason for the open world semantics of concept languages is that they are generally used in applications where incomplete information have to be accounted for. For example, even if all the known friends of John are male, one does not want to conclude that all friends of John are male.

On the other hand, there are situations where it is natural to query a knowledge base under the closed world assumption. It is important to note the difference between assigning a closed world semantics to the knowledge base and allowing one to pose queries under the assumption that part of the knowledge base is complete.

We argue that the use of epistemic operators as described in the previous sections is a natural way to achieve such a flexible way of interacting with the knowledge base. Referring to the knowledge base Σ_1 of Example 2.2, consider the following examples:

Query 9: $\forall$FRIEND.$\exists$LOVES.$\top$(john).
Answer: UNKNOWN.

Query 10: $\forall$KFRIEND.K$\exists$LOVES.$\top$(john).
Answer: YES.

Query 9 gets the answer UNKNOWN because there is a model of Σ_1 where john is related to an individual z and z is not an instance of the concept $\exists$LOVES.$\top$. On the other hand, the reading of Query 10 is as follows: is it true that for every individual z known to be related to john through the role FRIEND, it is known that $\exists$LOVES.$\top(z)$ holds Σ_1? It is easy to see that the answer to this query is YES.

The above example shows that the use of K allows one to pose queries to a knowledge base Σ under the assumption that Σ has complete knowledge on a certain individual a and a certain role P (john and FRIEND in the example), i.e. under the assumption that for every pair (a, b) such that $\Sigma \not\models P(a, b)$, $P(a, b)$ is false in Σ. Notice that this is not the same as assuming that the knowledge about every role is complete, like for example in [13].

In [11] the problem of translating Circumscription into epistemic theories is discussed. We show here that our query language allows us to achieve at least the expressive power of the (naive) Closed World Assumption (CWA) (see [15]). We do so by considering knowledge bases expressed in the simple language $\mathcal{AL}_0$, whose concepts are formed according to the rule:

$$C, D \longrightarrow A \mid \neg A \mid C \sqcap D \mid \forall R.C.$$

More complex languages and more powerful forms of closed world reasoning (e.g. Careful CWA) require a

more sophisticated treatment, which is outside the scope of this paper.

Let Σ be an $\mathcal{ALC}_0$-knowledge base, C be any $\mathcal{ALC}$-concept, and a be an individual. Σ entails $C(a)$ under the CWA, written $\Sigma \models_{CWA} C(a)$, if $C(a)$ is true in every minimal model of Σ.

Given a simple $\mathcal{ALC}$-concept C, we define the $\mathcal{ALCK}$-concept $\overline{C}$ as follows:

1. $\overline{A} = \mathbf{K}A$
2. $\overline{\neg A} = \neg \mathbf{K}A$
3. $\overline{(C \sqcap D)} = \mathbf{K}\overline{C} \sqcap \mathbf{K}\overline{D}$
4. $\overline{(C \sqcup D)} = \mathbf{K}\overline{C} \sqcup \mathbf{K}\overline{D}$
5. $\overline{(\exists P.C)} = \exists \mathbf{K}P.\overline{C}$
6. $\overline{(\forall P.C)} = \forall \mathbf{K}P.\overline{C}$.

Proposition 5.1 *Let Σ be an $\mathcal{AL}_0$-knowledge base, C be an $\mathcal{ALC}$-concept, and a be an individual. Then $\Sigma \models_{CWA} C(a)$ if and only if $\Sigma \mid\!\approx \overline{C}(a)$. Moreover, checking if $\Sigma \mid\!\approx \overline{C}(a)$ can be done in polynomial time.*

One interesting consequence of the above result is that epistemic queries allow us to achieve the expressive power of relational query languages. Let us call $\mathcal{ALCK}^+$ the language whose syntax is as follows (C, D denote concepts, R, Q denote roles, and a denotes an individual):

$$\begin{array}{rcl} C, D & \longrightarrow & A \mid \{a\} \mid C \sqcap D \mid C \sqcup D \mid \neg C \mid \forall R.C \mid \\ & & \exists R.C \mid \mathbf{K}C \\ R & \longrightarrow & P \mid Q \\ Q, Q_1 & \longrightarrow & \mathbf{K}P \mid Q \sqcap Q_1 \mid Q \sqcup Q_1 \mid Q \circ Q_1 \mid \neg Q \mid \\ & & Q^* \mid Q^{-1}, \end{array}$$

where $\{a\}$ denotes the concept whose extension is constituted by the single individual a (see Section 4), $\circ$ denotes role concatenation, Q^* denotes the transitive closure of Q, and Q^{-1} denotes the inverse of Q.

An $\mathcal{ALCK}^+$-query over a knowledge base Σ is an expression of the form

$$\langle x_1, \ldots, x_n \mid \psi \rangle$$

where ψ is a first order formula whose free variables are among $x_1, \ldots, x_n$, whose constants are symbols in $\mathcal{O}_\Sigma$, and whose atomic formulas are of the form $\mathbf{K}C(t)$ or $\mathbf{K}Q(s,t)$, where C is an $\mathcal{ALCK}^+$-concept, Q is an $\mathcal{ALCK}^+$-role or is the predicate symbol $=$, and s, t are either constants or variables. The answer set of the above $\mathcal{ALCK}^+$-query is the set of tuples $\langle a_1, \ldots, a_n \rangle$, where each $a_i \in \mathcal{O}_\Sigma$, such that $\Sigma \mid\!\approx \psi[x_1/a_1, \ldots, x_n/a_n]$. We are using here a straightforward extension of the $\mid\!\approx$ relation defined in Section 2 to the case where the right hand side argument is a formula. Obviously, a corresponding extension of the method described in Section 3 is needed in order to evaluate $\mathcal{ALCK}^+$-queries.

$\mathcal{ALCK}^+$-queries constitute a superset of relational calculus queries [19]. Indeed, let us denote with $rdb(\Sigma)$ the relational database obtained from the $\mathcal{AL}_0$-knowledge base Σ as follows. First, for every primitive concept A in Σ, we introduce a relation $rel(A)$ of arity 1, and for every role R in Σ, we introduce a relation $rel(R)$ of arity 2. Second, let $rel(A)$ consist of the tuples $\langle a \rangle$ such that $a\!:\!A$ is in the completion S_Σ,[6] and let $rel(R)$ consist of the tuples $\langle a, b \rangle$ such that aRb is in S_Σ. It is possible to show that, for every satisfiable $\mathcal{AL}_0$-knowledge base Σ, and for any relational calculus expression r over $rdb(\Sigma)$, there is an $\mathcal{ALCK}^+$-query q over Σ such that the set of tuples in the answer to r is equal to the answer set of q. This proves that $\mathcal{ALCK}^+$ has at least the expressive power of the relational calculus.[7]

Another interesting feature of $\mathcal{ALCK}^+$-queries is that they provide a formal setting for extracting information from a knowledge base where part of the knowledge is expressed in a concept language, and part is stored in a relational database.

A further aspect that is usually considered in databases but not in concept languages, is that of integrity constraints, which are sentences specifying the set of admissible database states. In [16] it is argued that integrity constraints are naturally viewed as epistemic sentences specifying what the knowledge base is supposed to know about a particular aspect of the world, rather than a direct property of the world.

For example, if the we want to rule out those knowledge bases which are uncertain about the sex of every person who is known to be student, we cannot simply state that every student has a sex. Rather, we need to specify that for every known student a, the knowledge base either knows that a is a male, or knows that a is a female.

In order to represent this idea in our setting, we propose to model integrity constraints as $\mathcal{ALCK}$ concepts.

Definition 5.2 *An integrity constraint is an $\mathcal{ALCK}$-concept. Given an $\mathcal{ALC}$-knowledge base Σ and an integrity constraint C, Σ satisfies C if for every individual $a \in \mathcal{O}_\Sigma$, $\Sigma \mid\!\approx C(a)$. If $IC = \{C_1, \ldots, C_n\}$ is a set of integrity constraints, then Σ is said to be legal with respect to IC if Σ satisfies C_i for each $i \in \{1, \ldots, n\}$.*

For example, the previous integrity constraint can be expressed as:

$$(\neg \mathbf{K}\texttt{Student}) \sqcup (\mathbf{K}\texttt{Male} \sqcup \mathbf{K}\neg\texttt{Male}).$$

[6] Note that, due to the language constructs of $\mathcal{AL}_0$, there exists exactly one completion of S_Σ.

[7] $\mathcal{ALCK}^+$ is actually more expressive, since transitive closures cannot be expressed in the relational calculus [19].

Notice that integrity constraints satisfaction (i.e. checking whether a knowledge base is legal w.r.t. a set of integrity constraints) can be easily realized through the calculus presented in Section 3.

6 Formalizing Trigger Rules

Up to now we have considered only knowledge bases constituted of assertions on individuals. Many practical systems for building knowledge based applications, such as CLASSIC [2], or CLASP [21], provide an additional mechanism, called *trigger rules*, for specifying the knowledge base. Roughly speaking, trigger rules have the form: "if an individual is proved to be an instance of C, then derive that it is also an instance of D," where C, D are concepts (see [2]). The behavior of trigger rules is usually described in terms of a forward reasoning process that adds to the knowledge base the assertion $D(a)$ whenever $C(a)$ is proved to hold. Let us call *procedural extension* of Σ w.r.t. to Γ the knowledge base resulting from such a forward reasoning process on a set of assertions Σ and a set of trigger rules Γ.

Trigger rules in the context of frame-based systems are often defined informally. Attempts to precisely capture the meaning of such rules are based either on viewing them as knowledge base updates (see for example the *TELL* operation of [10]), or on ad hoc semantics (see [17]).

Our aim in this section is to show that trigger rules can be nicely formalized in our setting. To this end, we regard the specification Φ of a knowledge base as constituted by a pair $\langle \Sigma, \Psi \rangle$, where Σ is a set of membership assertions in $\mathcal{ALC}$, and Ψ is a set of epistemic sentences (representing trigger rules), each one of the form $\mathbf{K}C \Rightarrow \mathbf{K}D$, where C and D are $\mathcal{ALC}$-concepts. An epistemic interpretation $(\mathcal{I}, \mathcal{W})$ satisfies the sentence $\mathbf{K}C \Rightarrow \mathbf{K}D$ if $(\mathbf{K}C)^{\mathcal{I},\mathcal{W}} \subseteq (\mathbf{K}D)^{\mathcal{I},\mathcal{W}}$. Extending the definition given in Section 2, an epistemic model of $\langle \Sigma, \Psi \rangle$ is a pair $(\mathcal{I}, \mathcal{W})$, where $\mathcal{I} \in \mathcal{W}$ and $\mathcal{W}$ is any maximal subset of $\mathcal{M}(\Sigma)$ such that for each $\mathcal{J} \in \mathcal{W}$, $(\mathcal{J}, \mathcal{W})$ satisfies every sentence in Ψ.

Intuitively, the set of epistemic sentences Ψ restricts the set of models of Σ to the maximal subsets that satisfy every sentence in Ψ. Because of the form of such sentences, it can be shown that there exists only one maximal subset $\mathcal{W}$ of $\mathcal{M}(\Sigma)$ such that for all $\mathcal{J} \in \mathcal{W}$, $(\mathcal{J}, \mathcal{W})$ satisfies every assertion in $\langle \Sigma, \Psi \rangle$. Moreover, it can be shown that such $\mathcal{W}$ coincides with the set of models of the procedural extension of Σ w.r.t. the trigger rules represented by Ψ.

Example 6.1 Consider the $\mathcal{ALC}$-knowledge base $\Phi = \langle \Sigma, \Psi \rangle$ (inspired by an example in [2]):

$$\Sigma = \{\texttt{EATS(susan,chips)}, \texttt{Student(susan)}\}$$
$$\Psi = \{\texttt{KStudent} \Rightarrow \texttt{K}(\forall \texttt{EATS.JunkFood})\}.$$

One can verify that for every epistemic model $(\mathcal{I}, \mathcal{W})$ of Φ, we have $\gamma(\texttt{susan}) \in (\forall \texttt{EATS.JunkFood})^{\mathcal{I},\mathcal{W}}$ and $\gamma(\texttt{chips}) \in \texttt{JunkFood}^{\mathcal{I},\mathcal{W}}$, i.e., both the assertion $\forall \texttt{EATS.JunkFood(susan)}$ and $\texttt{JunkFood(chips)}$ are logical consequences of Φ, as one would expect.

Notice that, since trigger rules are used only in one direction, they are not expressible by logical implications. Indeed, our formalization with the epistemic operator correctly captures this intuition. Consider for instance the knowledge base $\Phi' = \langle \{\neg B(a)\}, \{\mathbf{K}A \Rightarrow \mathbf{K}B\} \rangle$, and observe that there exists an epistemic model $(\mathcal{I}, \mathcal{W})$ of Φ' such that $\gamma(a) \notin \neg A^{\mathcal{I}}$. Therefore, $\neg A(a)$ is not a logical consequence of Φ'.

We now turn our attention to the problem of deciding whether the specification $\langle \Sigma, \Psi \rangle$ of a knowledge base is satisfiable or not. In order to provide a calculus for such a problem, we add to rules 1–4 of Section 3 a new propagation rule, dealing with the sentences in Ψ. The new propagation rule is as follows:

5. $S \rightarrow_{\Rightarrow} \{a:D\} \cup S$
   if $(\mathbf{K}C \Rightarrow \mathbf{K}D)$ is in Ψ, $a:D$ is not in S, and every completion of $S \cup \{a:\neg C\}$ contains a Σ-clash.

The above rule, together with rules 1–4 of Section 3, provides a sound and complete calculus for the satisfiability of $\langle \Sigma, \Psi \rangle$. Indeed, it can be shown that $\langle \Sigma, \Psi \rangle$ is satisfiable if and only if there exists at least one completion of $\langle \Sigma, \Psi \rangle$ without Σ-clash (note that the only possible Σ-clashes in this cases are either of type 1 or of type 2). Moreover, it can be shown that the calculus can be turned into an algorithm that again works in PSPACE.

In addition to the above method for checking the satisfiability of a knowledge base $\langle \Sigma, \Psi \rangle$, where Ψ is a set of trigger rules, a complete account of the procedural extension requires a method for the construction of a new knowledge base as a result of the application of a set of trigger rules. To this purpose it is possible to devise a refined version of the above rule, which incrementally adds the consequences of trigger rules to the knowledge base Σ. This modification is straightforward in principle and we do not present it here for lack of space.

7 Conclusions

In the present paper we have presented a framework for adding epistemic operators to concept languages. By enriching concept languages with an epistemic operator, to distinguish what is known to the knowledge base as opposed to what is true in the external world, we have been able to formally characterize several aspects of frame-based systems.

In particular, we have considered the use of the epis-

temic operator in the query language, and shown that this richer query language may be used to reduce the complexity of reasoning within knowledge bases. In addition, by virtue of the epistemic operator one can naturally specify forms of closed-world reasoning in the queries, as well as integrity constraints, and compare the query language with relational algebra. Finally, we have shown how to formalize procedural mechanisms such as trigger rules.

However, several aspects of frame-based systems still remain to be explored, for example representing object-oriented database structures, default properties of concepts, and rule-oriented knowledge structures.

Perhaps, the most important aspect of this work is that a single representation mechanism allows for the treatment of a large number of features, which seems encouraging if we are to bridge the gap between theoretical work on frame-based languages and implemented systems.

References

[1] BAADER, F., AND HOLLUNDER, B. A terminological knowledge representation system with complete inference algorithm. In *Proc. of the Workshop on Processing Declarative Knowledge, PDK-91* (1991), Lecture Notes in Artificial Intelligence, Springer-Verlag.

[2] BORGIDA, A., BRACHMAN, R., MCGUINNESS, D., AND RESNICK, L. CLASSIC: A structural data model for objects. In *ACM SIGMOD International Conf. on Management of Data* (1989), pp. 58–67.

[3] BRACHMAN, R. J., AND LEVESQUE, H. J. The tractability of subsumption in frame-based description languages. In *Proc. of the 4th Nat. Conf. on Artificial Intelligence AAAI-84* (1984).

[4] DONINI, F. M., LENZERINI, M., NARDI, D., AND NUTT, W. The complexity of concept languages. In *Proc. of the 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning KR-91* (1991), J. Allen, R. Fikes, and E. Sandewall, Eds., Morgan Kaufmann, pp. 151–162.

[5] DONINI, F. M., LENZERINI, M., NARDI, D., AND NUTT, W. Tractable concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence IJCAI-91* (Sidney, 1991).

[6] DONINI, F. M., LENZERINI, M., NARDI, D., AND SCHAERF, A. From subsumption to instance checking. Tech. rep., Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Forthcoming.

[7] DOYLE, J., AND PATIL, R. S. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence 48* (1991), 261–297.

[8] HOLLUNDER, B. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proc. of the German Workshop on Artificial Intelligence* (1990), Springer-Verlag.

[9] LENZERINI, M., AND SCHAERF, A. Querying concept-based knowledge bases. In *Proc. of the Workshop on Processing Declarative Knowledge, PDK-91* (1991), Lecture Notes in Artificial Intelligence, Springer-Verlag.

[10] LEVESQUE, H. J. Foundations of a functional approach to knowledge representation. *Artificial Intelligence 23* (1984), 155–212.

[11] LIFSCHITZ, V. On open defaults. In *Symposium on computational logics*, J. W. Lloyd, Ed. Springer-Verlag, ESPRIT Basic Research Action Series, 1990, pp. 96–113.

[12] LIFSCHITZ, V. Nonmonotonic databases and epistemic queries. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence IJCAI-91* (Sidney, 1991).

[13] NEBEL, B. *Reasoning and Revision in Hybrid Representation Systems.* Lecture Notes in Artificial Intelligence. Springer-Verlag, 1990.

[14] NEBEL, B. Terminological reasoning is inherently intractable. *Artificial Intelligence 43* (1990), 235–249.

[15] REITER, R. On closed world data bases. In *Logic and Databases*, H. Gallaire and J. Minker, Eds. Plenum, 1978, pp. 119–140.

[16] REITER, R. On asking what a database knows. In *Symposium on computational logics* (1990), J. W. Lloyd, Ed., Springer-Verlag, ESPRIT Basic Research Action Series, pp. 96–113.

[17] SCHILD, K. Towards a theory of frames and rules. Tech. rep., FB Informatik, Technische Universität Berlin, Berlin, Germany, 1989.

[18] SCHMIDT-SCHAUSS, M., AND SMOLKA, G. Attributive concept descriptions with complements. *Artificial Intelligence 48*, 1 (1991), 1–26.

[19] ULLMAN, J. *Principles of Database and Knowledge Base Systems*, vol. 1. Computer Science Press, Potomac, Maryland, 1988.

[20] WOODS, W. A. Understanding subsumption and taxomony: A framework for progress. In *Principles of Semantic Networks*, J. Sowa, Ed. Morgan Kaufmann, 1991, pp. 45–94.

[21] YEN, J., NECHES, R., AND MACGREGOR, R. CLASP: Integrating term subsumption sstems and production systems. *IEEE trans. on Knowledge and Data Engineering 3*, 1 (1991), 25–31.

V.

Natural Language Processing

A General Semantic Model of Negation in Natural Language: Representation and Inference

Łucja Iwańska
GE Research and Development Center
#1 River Rd, Schenectady, NY 12301, USA
iwanska@crd.ge.com
(518)387-6077

Abstract

I present a computational model of negation in natural language that allows a computer program to simulate logical inferences that people draw as the result of understanding sentences with explicit negative information. I demonstrate that Boolean properties of natural language can be successfully modeled with knowledge representation formalisms with Boolean semantics.

1 INTRODUCTION

Handling negation in natural language is crucial because negation dramatically affects reasoning, entirely different conclusions may follow from a sentence that involves negation than from the sentence that does not; and because negation abounds in both written and spoken language. I present a formal, computational model of general semantic properties of negation in natural language. This model allows the UNO [1] natural language processing system to simulate logical inferences that people draw as the result of understanding sentences with explicit negative information. My model demonstrates that Boolean properties of natural language can be successfully modeled with knowledge representation formalisms with Boolean semantics.

The problem of computationally handling negation in natural language has not been addressed systematically. No approach accounts for the general semantic properties of negation captured by my model. Previous research on the semantics of negation belongs to one of the following three categories:

1. **Propositional negation,**
2. **Negation as failure,**
3. **Others,** other approaches to negation.

[1] The acronym UNO derives from two words: *Un*ification, the main operation in the formalism presented in this paper, and *Neg O*, the Latin word meaning *deny*.

The limitation of modeling negation in natural language with a **propositional negation**, a truth-functional operator on logical formulas, is addressing only negation of sentences, while in natural language any content category can be negated. A simple operator mapping truth values into truth values cannot account for the properties of negation of such semantically complex categories as determiners, adjectives, or adverbs. Also, sentential negation accomplished with the operator *It is not the case that* is rather rare in natural language [Horn, 1989].

Propositional negation is present in **logic-based approaches to natural language processing.** Within the framework of **the unification-based approaches to grammar**, negation is investigated as a means of increasing the expressability of formalisms for describing constraints on grammatical relations [Kasper and Rounds, 1986] [Moshier and Rounds, 1987] [Dawar and Vijayashanker, 1989], [Johnson, 1990]. Negation is viewed as a propositional operator in a meta-language based on logic. One conclusion of this research was that negative constraints on partially specified data structures cannot be propagated via unification if negation is given a classical interpretation [Pereira, 1987]. This problem is known as the non-monotonicity of negation, or as failure of upward closure under subsumption.

The formalism presented in this paper can be used for describing and computing negative constraints. The data structures are partially specified, yet negative constraints can be propagated via unification. The problem of the non-monotonicity of negation does not seem to arise; the formalism can handle Pereira's example illustrating the problem [Iwańska, 1992].

Propositional negation is also present in **knowledge representation formalisms** because their semantics are usually based on logic [Brachman et al., 1985] [Bobrow and Winograd, 1985] [Schmolze, 1989]. There appears to be no knowledge representation formalism that would offer a semantically clean complement operator on the representation of concepts. An-

other limitation of the available knowledge representation formalisms is that they either do not handle quantifiers at all, or handle very few quantifiers, typically involving the determiners *some* and *every* only.

The second type of negation present in the literature, **negation as failure** [Clark, 1978], lacks semantics and goes beyond first-order reasoning [Reiter, 1985] [Lifschitz, 1987] [Geffner, 1989].

The negation of [Karttunen, 1984] is a binary operation on feature structures and lacks semantics.
[Nakazawa et al., 1988] offer two negations with different semantics, one for atomic values and one for complex features, which is counterintuitive. They do not specify how the unification of negated complex features should proceed.
[Padgham, 1989] characterized the complement operation in her Boolean lattice of types by a paraphrase of the complement law of a Boolean algebra. Inferencing in her inheritance network consists of traversing various links, including negative links, which means that reasoning lacks semantics.
[Schwartz, 1989] defined three different negations for scalar predicates such as *tall*, *short*, and *very short*. This proliferation of negations is unjustified because the underlying semantics of all scalar predicates is the same.
[Horty and Thomason, 1990] allow for complex Boolean combinations of properties of individuals in their inheritance networks. They do not provide semantics of their theory or its implementation. Another problem with their approach is that they replace negated nodes with 'positive' ones; for example, a property node *unoccupied* gets replaced with another property node *available*. This does not work in general; for example, it is not clear what should replace *not green*.

This paper is organized as follows: Section 2 discusses the scope of the proposed model; Section 3 presents one of the two knowledge representation formalisms on which the UNO representation of natural language is based; Section 4 discusses the UNO representation of natural language, the algorithm for representing knowledge derived from natural language sentences and drawing inferences, and the UNO natural language processing system; Section 5 sketches future work; Section 6 contains conclusions.

2 PROPOSED MODEL

2.1 CHARACTERISTICS OF NEGATION

My model accounts for the following general semantic properties of negation in natural language:

1. Expressions of eleven syntactic categories can be negated;
2. Negation of determiners and verb phrases reverses entailment of sentences;
3. Negation of a verb phrase may affect any part of the sentence;
4. Negation of scalar predicates has a *less than* interpretation.

The sentences in Table 1 illustrate the first property of negation in natural language.

Table 1: The UNO Representation Accounts for the Negation of Eleven Content Categories: Determiner, Adjective, Common Noun, Proper Noun, Adverb, Verb, Verb Phrase, Noun Phrase, Preposition, Prepositional Phrase, and Sentence.

Sentences with Explicit Negation
Not many dogs like cats
John is very *unhappy, indecisive*, but *not stupid*
Noncitizens have very few political rights
John, and *not Mary*, walks fast
Tommy is a very, but *not extremely*, small baby
Mary truly *dislikes* him
John *doesn't love Mary a lot*
John likes dogs, and *not cats*
He is *not in*, but near, Warsaw
John is in the garden, and *not in the park*
It is not the case that John walks fast

The second property of negation is illustrated by the following example. The sentence ***Every student works hard*** entails the sentence ***Every student works***. When the determiner ***every*** in these sentences is negated, the direction of entailment gets reversed: the sentence ***Not every student works*** entails the sentence ***Not every student works hard*** [Barwise and Cooper, 1981].

The third property of negation is illustrated by the sentence ***John doesn't love Mary a lot*** which may be interpretated as one of the following:

1. ***John loves Mary but less than a lot,***
2. ***John loves someone else,***
3. ***Someone else loves Mary,***
4. ***John likes, and not loves, Mary.***

The fourth property of negation is illustrated by the sentence ***Mary doesn't have many records*** which is usually understood as asserting that Mary has fewer records than many.

My logical negation corresponds to Horn's descriptive negation [Horn, 1989]. He distinguishes two types of negations— descriptive and metalinguistic. The descriptive negation corresponds to the typical reading of a sentence. In the sentence ***Mary doesn't have many records***, it produces the interpretation that Mary has

fewer records than many. Horn defines metalinguistic negation as 'a device for objecting to a previous utterance on any grounds whatever.' In some cases, metalinguistic negation has the same effect on logical inferences as descriptive negation; for example, the contrastive construction *John, and not Mary, is sleeping* entails the falsity of the sentence *Mary is sleeping*. My model accounts for such contrastive construction cases of metalinguistic negation.

2.2 REQUIREMENTS

My model fulfills the following three requirements:
1. Must mimic logical inferences in natural language;
2. Should parallel natural language;
3. Should be semantically clean.

The first requirement means that if a sentence is judged by speakers of English to entail another sentence, then the representation of these two sentences must preserve this entailment relation.

The model should parallel natural language in two aspects: 1) negation, conjunction, and disjunction in natural language should be modeled by the operations with provable semantics; 2) semantic properties attributed to a syntactic category as a class should be reflected by the representation.

For example, determiners as a class have a property of affecting the direction of entailment of sentences. Semantic peculiarities of the determiners *every* and *some* are responsible for the fact that the sentence *Every student works hard* entails the sentence *Every smart student works hard*, but when the determiner *every* is replaced with the determiner *some*, then the direction of entailment of these sentences is reversed: the sentence *Some smart student works hard* entails the sentence *Some student works hard*.

It is also the semantic properties of individual determiners that determine whether it is possible for a pair of sentences of the form $Det_1\,N\,VP$ and $Det_1\,N\,not\,VP$ to be true. For example, the determiner *many* is such that it is possible that *Many people like cats* is true and *Many people don't like cats* is true; the determiner *most* is such that it is impossible that *Most people like cats* is true and *Most people don't like cats* is true.

The third requirement means that data structures utilized by the model and the operations on these data structures should have provable semantics.

2.3 SUBSET OF ENGLISH

I investigate a substantial subset of English that involves sentences with complex expressions— quantified noun phrases with complex determiners, noun phrases involving proper nouns, complex common nouns modified by complex adjectives, complex verbs modified by complex adverbs, complex adjectives modified by complex adverbs, and Boolean combinations of expressions in all eleven syntactic categories.

2.4 FORMAL BASIS

I have chosen a complement operator in a Boolean algebra [Keenan and Faltz, 1985] as a formal, semantic model of negation in natural language because this mathematical model uniformly accounts for cross-categorial negation and captures the intuition that there is only one descriptive negation in natural language. Boolean properties of natural language make it desirable to have a natural language processing system whose underlying computational formalism is a Boolean algebra with the set-theoretic semantics. Sets underlie semantics of many syntactic categories of natural language: common nouns, verbs, and adjectives can be thought of as denoting sets of persons or objects that possess properties denoted by the words; adjective and adverbs are functions mapping sets of objects into sets of objects; determiners are functions mapping sets of objects into sets of sets of objects, and the denotations of proper nouns are sets of sets of objects [Dowty et al., 1981] [Barwise and Cooper, 1981] [Keenan and Faltz, 1985] [Hamm, 1989].

I propose a representation of natural language, the UNO representation, based on two knowledge representation formalisms with Boolean semantics (these formalisms will be referred to as *UNO formalisms*):
1. A Boolean algebra of ϵ-terms[2],
2. Qualitative scales.

The Boolean operations of the UNO formalisms, meet, join, and complement, simulate conjunction, disjunction, and negation in natural language.

The formalism of the Boolean algebra of ϵ-terms builds upon [Aït-Kaci, 1984], a semantically clean version of semantic-network-style knowledge representation. I extended his lattice of ϵ-terms to a Boolean algebra by constructing a complement operator on ϵ-terms. The resulting formalism is a knowledge representation formalism with Boolean, set-theoretic semantics. Given explicit representations of concepts, it can compute explicit representations of their conjunctions, disjunctions, and complements.

The formalism of qualitative scales, a formalization of quantitative scales [Horn, 1989], captures the semantics of scalar predicates, expressions of natural language that denote certain quantitative values. Examples of scalar predicates are the determiner *many* and verb *like*. Scalar predicates are interesting because negation of scalar predicates has a 'less-than' interpretation and because the entailment relation holds between concepts that are denotations of lexically simple items, eg. *love* entails *like*— if one loves cookies,

[2]The notion of ϵ term is defined in Section 3.

then one also likes cookies. The formalism of qualitative scales will not be described here; I refer readers to [Iwańska, 1992].

My approach to automating inferencing in natural language is similar to Montague's [Dowty et al., 1981] in that I translate natural language sentences into a semantically clean representation, and model inferences by performing the operations with provable mathematical properties on this representation.

2.5 LIMITATIONS OF THE MODEL

There are a number of limitations of my model that result from the necessary and deliberate idealizations of the problem of negation in natural language. I only marginally address metalinguistic negation. I ignore pragmatics: 1) I do not distinguish between negative prefixes such as *dis* or *un*; 2) I offer a marginal account of the interaction of the scope of negation and quantifiers[3]. 3) I ignore the problem of presupposition. My model covers a limited subset of English. I ignore temporal information in English sentences. I do not address intensional properties of natural language or the problem of non-logical reasoning.

3 BOOLEAN ALGEBRA OF ϵ-TERMS

3.1 LATTICE OF ϵ-TERMS

[Aït-Kaci, 1984] formalized and operationalized semantic networks. He constructed a complete, distributive lattice of finite sets of first-order terms, called *ϵ-terms*. Semantically, ϵ-terms are sets of objects in the interpretation universe, and the lattice operations, $\sqcap_\epsilon$ (ϵ-meet) and $\sqcup_\epsilon$ (ϵ-join), compute terms whose denotations are their set-intersections and set-unions. The operations $\sqcap_\epsilon$ and $\sqcup_\epsilon$ induce an ordering relation $\sqsubseteq_\epsilon$ on terms which reflects term subsumption, semantically, set-containment.

```
[ a(k => v1: b,
    m => a(k => v1)) ]
```

Figure 1: An ϵ-Term T with a Single ψ-Term t

An ϵ-term consists of a finite number of elements called *ψ-terms* (see Figure 1). A ψ-term improves the notion of a PROLOG term [Sterling and Shapiro, 1986] [Aït-Kaci and Nasr, 1985] in that neither the number nor the order of its attributes is fixed, and atomic head symbols are unifiable. A ψ-term is a record-like structure: it has a *head* which is a primitive type symbol, and a *body* which is a possibly-empty list of attribute-value pairs uniquely associating labels with terms. Semantically, primitive types are sets of objects in the interpretation universe; the smallest type is *false*, and the largest *true*; $\wedge$, $\vee$, and $\neg$ is the meet, join, and complement operation on primitive types. Labels are functions mapping sets of objects into sets of objects. Addresses of a ψ-term are ordered sets of its labels.

For example, the ψ-term t in Figure 1, has the type a as its head, two attributes with labels k and m; the value of the first attribute is a primitive type b, and the value of the second attribute is a ψ-term with the primitive type a as its head and one attribute with label k and value b; the ψ-term t has three explicit addresses: $\{k\}$, $\{m\}$, and $\{m, k\}$; variables mark addresses that corefer (coreference is structure sharing); the variable $v1$ marks the coreference of the addresses $\{k\}$ and $\{m, k\}$. Semantically, the ψ-term t stands for the subset of objects of type a on which function k returns objects of type b and function m returns objects in the denotation of $a(k => b)$; coreference further constrains this set to those objects for which both functions k return the same objects.

$T_1 = [a(k => a),\ b] \qquad T_2 = [b(m => a)]$

$T_1 \sqcup_\epsilon T_2 = [a(k => a),\ b,\ b(m => a)] = [a(k => a),\ b] = T_1$

$T_1 \sqcap_\epsilon T_2 = [(a \wedge b)(k => a, m => a),\ b(m => a)] = [b(m => a)] = T_2$

$not_\epsilon T_1 = [\neg a,\ a(k => \neg a)] \sqcap_\epsilon [\neg b] = [(\neg a \wedge \neg b),\ (a \wedge \neg b)(k => \neg a)]$

Figure 2: Examples of Boolean Operations on ϵ-Terms

The operation $\sqcup_\epsilon$ on ϵ-terms deletes subsumed elements from the union of the elements of the terms. The operation $\sqcap_\epsilon$ on ϵ-terms deletes subsumed elements from the pairwise unification of the elements of the terms. Figure 2 shows examples of these operations. The sample terms are in the subsumption relation $T_2 \sqsubseteq_\epsilon T_1$, i.e., T_1 subsumes T_2 (T_2 entails T_1).

Subtyping relation can be defined and computed via a set of type equations. For example, the fact that cats and dogs are animals, and that dobermans are dogs, and therefore dobermans are also animals, can be encoded by the following two equations

```
animal  ==>  [ cat, dog ] .
dog     ==>  [ doberman ] .
```

I will sometimes omit subscripts on the operations; in the expression *not* R, if R is a primitive type, then *not* is $\neg$, if R is an ϵ-term, then *not* is not_ϵ.

[3] [Horn, 1989] argues convincingly that the problem of determining the relative scope of negation and quantifiers is pragmatic, and not logical. In [Iwańska, 1992], I have formulated a more adequate criterion of the occurence of the wide scope of negation than the criterion offered in [Horn, 1989].

3.1.1 Negative Information

Aït-Kaci sketched two ways of handling negative information in his calculus: inequalities among subterms to be built into the unification algorithm and complemented types. A complemented type $t_1 \setminus t_2$ specifies anything that subsumes t_1 but does not subsume t_2.

There are a number of problems with handling negative information via such complemented types. Complemented types do not produce an explicit complement term. Complemented types do not parallel natural language: 1) they are functions of two arguments, yet negation in natural language is an unary operator; 2) they stand for complex Boolean expressions; for example, the interpretation of the complemented type (*animal* \ *cat*) is defined as those elements of the type *animal* that are not *cat*, which corresponds to a complex Boolean expression *animal and not cat*. Complemented types are sketched only. The join operation on complemented types is not defined and one cannot mimic natural language expressions that involve both negation and disjunction. Inequalities among subterms is a special case of type complementation. For example, the complement of a ψ-term $a(l \Rightarrow v1 : b, k \Rightarrow v1)$ should contain a ψ-term $a(l \Rightarrow b, k \Rightarrow b)$ such that the subterms at the addresses $\{l\}$ and $\{k\}$ are not equal.

3.2 ϵ-COMPLEMENT

My ϵ-complement operator solves the problems with the complemented types of Aït-Kaci. Given an ϵ-term T, this unary operator with Boolean, set-complement semantics produces an explicit ϵ-term $not\,T$ such that no element in the denotation of T belongs to the denotation of $not_\epsilon T$, i.e., $T \sqcap_\epsilon not_\epsilon T = [\,false\,]$; at the same time, $T \sqcup_\epsilon not_\epsilon T = [\,true\,]$.

If $T = \bigcup_{i=1}^{n} \{t_i\}$ is an ϵ-term, then its complement is defined as $not_{\epsilon_r}(T) = \sqcap_{\epsilon_r i=1}^{n} not(t_i)$, i.e., it is the result of the $\sqcap_\epsilon$ operation on the ϵ-terms that are complements of the elements of T. Figures 2 and 4 illustrate ϵ-complement for the terms T_1 and T.

$t^{max}(t) =$	$not(t^{max}(t)) =$

a(k => b,
  m => a(k => b))
	[not a,
	  a(k => not b),
	  a(m => not a),
	  a(m => a(k => not b))]

Figure 3: The Maximum Term of the ψ-Term t in Figure 1 and its Complement

The complement operator on ψ-terms separately negates two orthogonal constraints that a ψ-term represents: 1) type constraints, and 2) coreference constraints. For the term t in Figure 1, an example of the type constraint is the fact that only elements of type a belong to the denotation of t; an example of the coreference constraint in t is the fact that both functions k return the same objects. In a complement term, at least one of these orthogonal constraints does not hold. I separate type constraints from coreference constraints by relaxing the coreference constraint. The operation of relaxing coreference constraints in a ψ-term results in another ψ-term, called *maximum term*, such that it has the same addresses and type constraints as the original term but no two addresses corefer (see Figure 3).

```
[ not a,
  a(k => not b),
  a(m => not a),
  a(m => a(k => not b)),

  a(k => b,                  [ a(k => v1: b
    m => a(k => b))  \          m => a(k => v1)) ]  ]
```

Figure 4: A Complement of the ϵ-Term T in Figure 1

The complement of the maximum term of a ψ-term represents negating type constraints only. I define a complement operation on such maximum terms as follows: for each address in the maximum term, I construct the largest (according to the subsumption relation) term built around this address, and replace the type at this address by its complement; each such new term has the same type constraints as the original maximum term, except for the type at the address it is built around; Figure 3 shows the complement of the maximum term of the ψ-term t.

There is no simple way of representing negation of coreference constraints. I use for this purpose complemented types of Aït-Kaci. A complemented type $t \setminus [t]$, where t is a ψ-term, represents the fact that no object satisfying the coreference constraints of the term t belongs to the interpretation of this complemented type. Such complemented types will appear in a term only as the result of negation.

I prove [Iwańska, 1992] that such an unary operator on ϵ-terms possesses the necessary Boolean properties [Keenan and Faltz, 1985]. As a consequence, this operator reverses the direction of the ordering relation in the algebra: $T_1 \sqsubseteq_{\epsilon_r} T_2$ iff $not_{\epsilon_r}(T_2) \sqsubseteq_{\epsilon_r} not_{\epsilon_r}(T_1)$.

The cost of computing the complement of a singleton ϵ-term is polynomial in the size of its element (the number of coreference classes of a term). In general, this cost is exponential due to the disjunctive nature of ϵ-terms.

4 THE UNO REPRESENTATION OF NATURAL LANGUAGE

4.1 COMMON NOUNS, VERBS, AND ADJECTIVES

I represent a lexically simple common noun, verb, and adjective by a singleton ϵ-term whose only element is a primitive type. For example, a common noun *baby*, the verb *walk*, and the adjective *red* are represented by the following ϵ-terms:

```
[ baby ]    [ walk ]    [ red ]
```

The UNO representation provides a clean, formal way to define and compute two types of entailment relations between lexically simple items. The first, exemplified by the entailment between *dog* and *animal*, can be defined and computed by the formalism described in Section 3. The second, the entailment relation between the elements of a qualitative scale, for example, the entailment between scalar verbs *like* and *love*, can be defined and computed by the formalism of qualitative scales [Iwańska, 1992].

Complex common nouns, verbs, and adjectives are represented by ϵ-terms with modifiers: their labels are functions that modifiers stand for. For example, a complex verb *walk fast* is represented by the term

```
[ walk(speed => fast) ]
```

The UNO representation of complex common nouns, verbs, and adjectives reflects their semantics and mimics the human intuition about the entailment relation between simple and complex expressions. If a woman *walks fast*, then she also *walks*, and the term representing *walks fast* entails the term representing *walks*. If a woman is *walking fast*, then she is not *walking slowly*. The UNO representation accounts for this fact because the scalars *fast* and *slowly* are inconsistent.

Adjectives modified by adverbs pose a problem. I want to represent a complex noun *very small baby* by the term

```
[ baby(size => small(adv => very)) ]
```

The human intuition about the relation between *very small baby* and *small baby* is mimicked: if something is a *very small baby*, then it is also a *small baby*, and the representation parallels this entailment. Similarly, the representation mimics the fact that if something is an *extremely small baby*, then it is also a *very small baby* because the scalar adverb *extremely* entails the scalar adverb *very*. The result of negating the property of *very small baby* also matches intuition. If something is not a *very small baby*, then it is a small, but not very small, baby, or it is a baby that is not small, or it is not a baby at all. The complement of the term representing *very small baby* contains three elements that give just that:

```
[ baby(size => small(adv => not very)),
  baby(size => not small),
  not baby ]
```

The problem is that the adverb *very* is a function and does not stand for a set of objects. I will allow for this semantic impurity in the UNO representation because satisfying the requirement *Must mimic logical inferences in natural language* takes precedence over satisfying the requirement *Should reflect semantics of different syntactic categories of natural language.*

4.2 NOUN PHRASES, DETERMINERS

I represent noun phrases involving proper nouns by ϵ-terms with Boolean combinations of primitive types. For example, a proper noun *Mary* and a noun phrase *Mary or Mark* are represented by the following terms:

```
[ mary ]      [ mary, mark ]
```

Representing quantified noun phrases involves representing determiners. Determiners are scalars [Horn, 1989] [Iwańska, 1992] whose ordering relation (entailment) is in general context-dependent [Barwise and Cooper, 1981]. The formalism of the Boolean algebra of ϵ-terms allows one to encode the semantics of first-order definable determiners. For example, the semantics of the determiners *every* and *some* are encoded by the following type equations

```
set ==> [ empty-set, cons ] .
dets ==> true(sem => set_meet(noun     => set,
                              verb     => set,
                              set_meet => set)) .
every ==> dets(sem => set_meet(noun     => v1:set,
                               set_meet => v1)) .
some ==> dets(sem => set_meet(set_meet => cons)) .
```

The type *set* allows one to represent sets in linked list notation. Both *every* and *some* are instances of the type *dets* (determiners). The semantics of *every* in a quantified sentence S = Det N VP is such that the intersection (set_meet) of the denotation of N and the denotation of VP must be the denotation of N itself [Barwise and Cooper, 1981]. This is encoded as the coreferentiality between the *noun* and the *set_meet* attributes. The semantics of *some* is such that the resulting intersection must be nonempty, or *cons* in the linked list representation of sets.

The UNO representation of the semantics of determiners allows one to compute the entailment relation on determiners by performing the ϵ-operations on their representations. For example, it can be computed that if the denotation of a noun is not empty, then *every* entails *some*, and that *not some* entails *not every*. The ordering relation on determiners gives a basis for

computing entailment of sentences with quantified expressions. For example, if the denotation of the noun *student* is not empty, then the sentence *Every student walks* entails the sentence *Some student walks*, and the sentence *No student walks* entails the sentence *Not every student walks*.

In the representation of noun phrases, determiners are represented by ϵ-terms. For example, the determiner *not very many* is represented by the term

```
not many(adv => very)
```

I represent a quantified noun phrase of the form *Det Noun* by the term

```
np(det => T(Det),
   n   => T(Noun))
```

where $T(Det)$ and $T(Noun)$ are terms representing the determiner *Det* and noun *Noun*.

Quantifiers in natural language are denotations of noun phrases [Barwise and Cooper, 1981] [Hamm, 1989]. **Monotonicity** of a quantifier defines the direction of entailment of sentences involving this quantifier in the subject noun phrase position. **Left monotonicity** defines the direction of entailment when nouns in subject noun phrases are in the entailment relation, and **right monotonicity** when verb phrases of the sentences are in the entailment relation.

[Barwise and Cooper, 1981] and later [Hamm, 1989] observed that internal negation (roughly, negation of the verb phrase) and external negation (roughly, negation of the determiner in the subject noun phrase position) of a quantifier reverse its right monotonicity, and that internal negation of a quantifier preserves and external negation reverses its left monotonicity. The UNO representation accounts for these two facts. The reversal of left monotonicity follows from reversing the direction of entailment between the representation of determiners, and the reversal of right monotonicity follows from reversing the direction of subsumption between verb phrases[4]. Determiners are scalars whose direction coincides with the right monotonicity of quantifiers [Iwańska, 1992]. The UNO system cannot automatically determine the direction of the right monotonicity of quantifiers involving proper nouns[5] or the direction of the left monotonicity. I assume that they are given as part of input.

4.3 VERB PHRASES

I represent verb phrases by ϵ-terms. For example, the verb phrases *love Mary* and *love Mary a lot* are represented by the terms

```
[ love(np => mary) ]    [ love(np  => mary,
                               adv => a_lot) ]
```

If one loves Mary a lot, then one also loves her. The representation matches the intuition: the term representing *love Mary a lot* entails the term representing *love Mary*. The negated verb phrases *not love Mary* and *not love Mary a lot* are represented by the terms

```
not [love(np => mary)]    not [love(np  => mary,
                                    adv => a_lot)]
```

If one does not love Mary, then one also does not love her a lot. Again, the representation matches this intuition: the term representing *not love Mary* entails the term representing *not love Mary a lot*.

The UNO representation of verb phrases captures the property of negation of affecting any part of the sentence, here any part of the verb phrase. The complement of the term representing the verb phrase in the sentence *John doesn't love Mary a lot* contains the elements that correspond to those different interpretations of the sentence:

```
[ not love,
  love(np  => not mary),
  love(adv => not a_lot) ]
```

The first element corresponds to the interpretation *like and not love* because the scalar *like* is consistent with the scalar *not love*; the second element corresponds to the interpretation *love someone else*; the third element corresponds to the interpretation *loves, but less than a lot*.

Representing verb phrases as ϵ-terms provides a computational vehicle for accounting for one of the characteristic properties of negation in natural language, that it reverses entailment of sentences.

4.4 SENTENCES

A declarative sentence asserts that an individual or a set of individuals possesses certain properties. For example, the sentence *John doesn't walk very fast* asserts that John has a *not walk very fast* property. A sentence whose subject noun phrase is a negated proper noun asserts that the denotation of this proper noun contains a property complementary to the property described by the verb phrase [Barwise and Cooper, 1981]. For example, the sentence *John and not Mary walks* asserts that Mary possesses a property of *not walking*.

I utilize Montague's uniform treatment of proper nouns and quantified noun phrases [Dowty et al., 1981]: their denotations are sets of properties expressed in natural language by nouns, verbs, and adjectives which are naturally encoded as sets of ϵ-terms. I represent sentences by knowledge base rules defined below.

[4] The representation of verb phrases is shown in the next section.

[5] [Barwise and Cooper, 1981] also stipulate the direction of the monotonicity of quantifiers involving proper nouns.

Definition 1 (Knowledge Base Rule)
NP --> { $P_1, \ldots, P_n$ } is a knowledge base rule, where
NP is the UNO representation of a proper noun, a disjunction of noun phrases, or a quantified noun phrase, P_i are ϵ-terms representing properties that the denotation of NP is known to possess.

The sentence *Every student works hard* is represented by the KB rule

```
np(det => every,
   n   => student) --> { [ work(adv => hard) ] }
```

This rule encodes the fact that the set of individuals denoted by the quantified noun phrase *Every student* possesses a single property of *working hard*.

4.5 INFERENCE

Both UNO formalisms are used to represent sentences and compute their truth-values by computing the entailment relation between the representations of their subject noun phrases and verb phrases. In case of quantified noun phrases, I utilize the fact that the entailment relation on determiners is directly relevant to the entailment of sentences. The algorithm for representing and updating knowledge derived from actual sentences uses the concepts defined below.

Definition 2 (NP-stronger than or equal to)
Let $S_1 = NP_1\, Verb_1$ be a sentence, NP_1 and NP_2 be

1. *either Boolean combinations of proper nouns such that $T(NP_1)$ and $T(NP_2)$ are their UNO representations, or*
2. *quantified noun phrases $NP_1 = Det_1\, Noun_1$ and $NP_2 = Det_2\, Noun_2$ such that $T(Det_1)$, $T(Det_2)$, $T(Noun_1)$, and $T(Noun_2)$ are terms representing Det_1, Det_2, $Noun_1$, and $Noun_2$ respectively*

NP_1 is np-stronger than or equal to NP_2 iff

1. *NP_1 is left monotone increasing, and*
   (a) *for proper nouns, $T(NP_1) \sqsubseteq_\epsilon T(NP_2)$*
   (b) *for quantified noun phrases, $T(Det_1) \sqsubseteq_\epsilon T(Det_2)$, and $T(Noun_1) \sqsubseteq_\epsilon T(Noun_2)$*
2. *NP_1 is left monotone decreasing, and*
   (a) *for proper nouns, $T(NP_2) \sqsubseteq_\epsilon T(NP_1)$*
   (b) *for quantified noun phrases, $T(Det_2) \sqsubseteq_\epsilon T(Det_1)$, and $T(Noun_2) \sqsubseteq_\epsilon T(Noun_1)$*

Definition 3 (NP-weaker than or equal to)
NP_1 is np-weaker than or equal to NP_2 if and only if NP_2 is np-stronger than or equal to NP_1.

In the sentence *Some bright students don't walk*, the term representing *some bright students* is **np-stronger than or equal to** the term representing *some students* because *some* is left monotone increasing, $[some] \sqsubseteq_\epsilon [some]$, and $[student(adj => bright)] \sqsubseteq_\epsilon [student]$.

Definition 4 (V-stronger than or equal to)
Let $S_1 = NP_1\, Verb_1$ be a sentence, $Verb_1$ and $Verb_2$ be verbs with $T(Verb_1)$ and $T(Verb_2)$ as their UNO representations.
$Verb_1$ is v-stronger than or equal to $Verb_2$ iff
1) NP_1 is right monotone increasing and $T(Verb_1) \sqsubseteq_\epsilon T(Verb_2)$, or
2) NP_1 is right monotone decreasing and $T(Verb_2) \sqsubseteq_\epsilon T(Verb_1)$

In the sentence *Many students don't walk*, the term representing *don't walk* is **v-stronger than or equal to** the term representing *don't walk fast* because *many* is right monotone increasing, and $[not\, walk] \sqsubseteq_\epsilon not\, walk(adv => fast)$.

Definition 5 (Relevant Property) *Let*

1. *T be an ϵ-term*
2. *H_T be an ϵ-term such that its elements are head types of the elements of T*
3. *NP_{KB} --> { ... P ... } be a KB rule*
4. *H_P be an ϵ-term such that its elements are head types of the elements of P*

Property P is relevant for T iff
$H_T \sqcap_\epsilon H_P \neq [false]$, or $H_T \sqcup_\epsilon H_P = [true]$.

The property [*work(adv => hard)*] is relevant for [*work, sing(adv => loud)*] because
$H_T \sqcap_\epsilon H_P = [work, sing] \sqcap_\epsilon [work] = [work] \neq [false]$.

Definition 6 (Updating Operation)
Let $S = NP_1\, Verb_1$ be a sentence,
NP_{KB} --> { ... P ... } a KB rule.

The updating operation for $Verb_1$ and a property P relevant for $Verb_1$ is
$\sqcap_\epsilon$ if NP_1 is right monotone increasing
$\sqcup_\epsilon$ if NP_1 is right monotone decreasing

If the input sentence is *John doesn't walk* and the KB contains the following rule

```
john --> { not walk(adv => fast) }
```

then the updating operation is $\sqcap_\epsilon$.

The UNO system uses the following algorithm for representing and applying knowledge derived from actual natural language sentences.

Processing a True Sentence: An Algorithm for Updating Knowledge Base

Input:

1. A true sentence S whose subject noun phrase and verb phrase are represented by the terms *New_NP* and *New_Verb* respectively; if the subject noun phrase is quantified, then *New_NP* is the term *np(det => New_Det, noun => New_Noun)*
2. Existing knowledge base KB (a set of KB rules)

Step 1: Check if S is entailed by KB

1. Find a KB rule NP -- > { P_1, ..., P_n } such that NP is np-stronger than or equal to the *New_NP*
   (a) If one of the properties $P_1 \ldots P_n$ is v-stronger than or equal to the *New_Verb*, then go to **Step 3** (S is entailed by KB)
   (b) If one of the properties $P_1 \ldots P_n$, say P_i, is relevant to *New_Verb* and $P_i \sqcap_e$ *New_Verb* = [*false*], then go to **Step 3** (S contradicts KB)
2. If no such rule is found, then
   (a) If *New_NP* is quantified, then compute the truth-value of S according to the semantics of the determiner *New_Det*:
      i. Compute the denotations of *New_Noun* and *New_Verb*: they are sets of those individuals that possess a property subsumed by the *New_Noun* and *New_Verb* respectively.
      ii. If the semantic condition of the determiner is satisfied, then go to **Step 3** (S is entailed by KB).
      Otherwise go to **Step 2**.
   (b) Otherwise go to **Step 2**.

Step 2: Update the KB

1. Find KB rules R = NP -- > { P_1, ..., P_n} such that
   - NP is np-weaker than or equal to *New_NP*
   - one of the properties is relevant for *New_Verb*
2. If no such R is found, then add a new rule *New_NP* -- > { *New_Verb* } to KB, and go to **Step 3**.
   Otherwise process each rule R individually:
   Replace each property P_i relevant for *New_Verb* with the result of the updating operation on P_i and *New_Verb*. If for any two rules the updated properties are in the v-stronger-than-or-equal-to relation, then remove the property from the rule whose left-hand side is np-weaker than or equal to *New_NP*.

Step 3: Output updated KB

The examples below illustrate the algorithm. Each example consists of a set of sentences to be processed in a given order, and a set of queries to the knowledge base. After processing each subsequent sentence, I will show both the updated knowledge base and the answers to the queries computed by the UNO system.

4.5.1 Sentences Whose Noun Phrases Involve Proper Nouns

Input sentences:

S_1 = *John and not Mary walks*
S_2 = *John sings or speaks*
S_3 = *John walks fast but speaks slowly*
S_4 = *John doesn't walk very fast*

Queries to the KB:

Q_1 = *Does John walk fast ?*
Q_2 = *Does Mary walk fast ?*
Q_3 = *Does John sing ?*
Q_4 = *Does John sing or speak ?*
Q_5 = *Does John walk very fast ?*

After processing S_1, KB_1 contains two rules

```
john    -->  { [ walk ] }
mary    -->  { [ not walk ] }
```

Q_1	Q_2	Q_3	Q_4	Q_5
Unknown	No	Unknown	Unknown	Unknown

After processing S_2, KB_2 contains two rules

```
john --> { [ walk ], [sing, speak] }
mary --> { [not walk] }
```

Q_1	Q_2	Q_3	Q_4	Q_5
Unknown	No	Unknown	Yes	Unknown

After processing S_3, KB_3 contains two rules

```
john    -->  { [ walk(adv => fast) ],
               [ speak(adv => slowly) ] }
mary    -->  { [ not walk ] }
```

Notice that the old property *sings or speaks* is replaced by a stronger property *speaks slowly*.

Q_1	Q_2	Q_3	Q_4	Q_5
Yes	No	Unknown	Yes	Unknown

After processing S_4, KB_4 contains two rules

```
john --> { [ walk(adv => fast(adv => not very)) ],
           [ speak(adv => slowly) ] }
mary --> { [ not walk ] }
```

Q_1	Q_2	Q_3	Q_4	Q_5
Yes	No	Unknown	Yes	No

4.5.2 Sentences Whose Noun Phrases Involve First-Order Definable Determiners

Input Sentences:

S_1 = *Every smart student works*
S_2 = *Every student works hard*

Queries to the KB:

Q_1 = *Does every smart student work hard ?*
Q_2 = *Does every student work ?*
Q_3 = *Does every student work hard ?*

After processing S_1, KB_1 contains one rule

```
np(det => every,
   n   => student(adv => smart)) --> { [ work ] }
```

Q_1	Q_2	Q_3
Unknown	Unknown	Unknown

After processing S_2, KB_2 contains one rule

```
np(det => every,
   n   => student) --> { [ work(adv => hard) ] }
```

Notice that the left-hand side of the KB rule is updated.

Q_1	Q_2	Q_3
Yes	Yes	Yes

4.5.3 Sentences Whose Noun Phrases Involve Determiners that are not First-Order Definable

Input Sentences:

S_1 = *Not very many dogs like cats*
S_2 = *Not many dogs like cats*
S_3 = *Not many dogs love cats*

Queries to the KB

Q_1 = *Is it true that not many dogs like cats ?*
Q_2 = *Is it true that not many dogs love cats ?*

After processing S_1, KB_1 contains one rule

```
np(det => not many(adv => very),
   n   => dog)
-->
{ [ like(np => cat) ] }
```

Q_1	Q_2
Unknown	Unknown

After processing S_2, KB_2 contains one rule

```
np(det => not many,
   n   => dog)      --> { [ like(np => cat) ] }
```

Q_1	Q_2
Yes	Yes

After processing S_3, KB_2 remains unchanged because it entails this sentence.

In the UNO representation of natural language sentences, only lower boundaries of a finite number of properties are explicitly stored in the KB. Each explicit property accounts for an infinite number of properties that an individual or a set of individuals possesses, which accounts for an infinite number of sentences entailed by the KB. For example, the UNO representation of a single property *walk extremely fast* accounts for the properties *walk very fast*, *walk fast*, *walk*, *walk or dance*, etc.; at the same time, this single property accounts for the lack of the properties *not walk*, *walk fast but not very fast*, etc.

4.5.4 Computing Truth-Values of Sentences

Given a sentence S, the UNO system can assign its truth-value: the sentence S is true if it is entailed by the KB; the sentence S is false if it contradicts the KB; the truth-value of the sentence S is otherwise unknown (the KB does not contain enough information to assign S a truth value).

The UNO system cannot always determine the truth-value of sentences involving determiners that are not first-order definable because it cannot represent their semantics explicitly. It can, however, account for some inferences. For example, after processing the sentence *Very many students don't walk*, UNO can infer that the sentence *Many students don't walk fast* is true.

4.5.5 Contradictions

The UNO algorithm system identifies sentences that contradict the existing knowledge base, but at this point, such sentences are ignored. However, it is an important feature of the UNO representation that contradictions admitted in the KB remain local. For example, given the contradictory KB below

```
john --> { [ walk ], [ not walk ], [ student ] }
mary --> { [ not walk ] }
bob  --> { [ dress(adv => well) ] }
```

the UNO system will be confused about the truth value of the sentence *John walks*, but it will correctly reason about the other properties of John, and about the properties of Mary and Bob.

4.6 IMPLEMENTATION

My model of negation is implemented as part of the UNO natural language processing system (see Figure 5) written in COMMON LISP. Three modules, **Reader**, **Dictionary**, and **Parser**, are modules of the BILING system [Iwańska, 1989].

The **Reader module** contains functions breaking input text into documents, paragraphs, sentences, and words. The **Dictionary module** contains functions creating, updating, loading, and checking consistency of the 2700 root English dictionary, and functions performing morphological analysis. The **Parser module** contains functions implementing a chart parser that produces both syntactic parse trees and the UNO semantic representation. The **Knowledge Representation module** consists of the **Boolean algebras module** and the **Knowledge Base Interpreter**. The **Boolean algebras module** implements

the UNO knowledge representation formalisms and some standard Boolean algebras such as predicate calculus and powerset of a set. This module also contains functions deriving the disjunctive normal form of a complex Boolean expression independently of its algebra, and functions creating the representation of the element in a particular algebra that a complex Boolean expression stands for. The **Knowledge Base Interpreter** implements the interpreter of the sets of type equations described in Section 3.1. The **Inference Engine** module implements the algorithm discussed in Section 4.5.

Embedding reasoning with explicit negative information into the system capable of handling real linguistic data did not degrade its performance. Processing examples discussed in Section 4.5 on a SPARC-1 workstation took 0.4 seconds of the System Run Time. This time includes: reading input file, breaking the input into documents, paragraphs, sentences, and words, performing morphological analysis of every word, producing syntactic parsing and the UNO semantic representation for each sentence and query, creating and updating dynamic knowledge base, and answering queries.

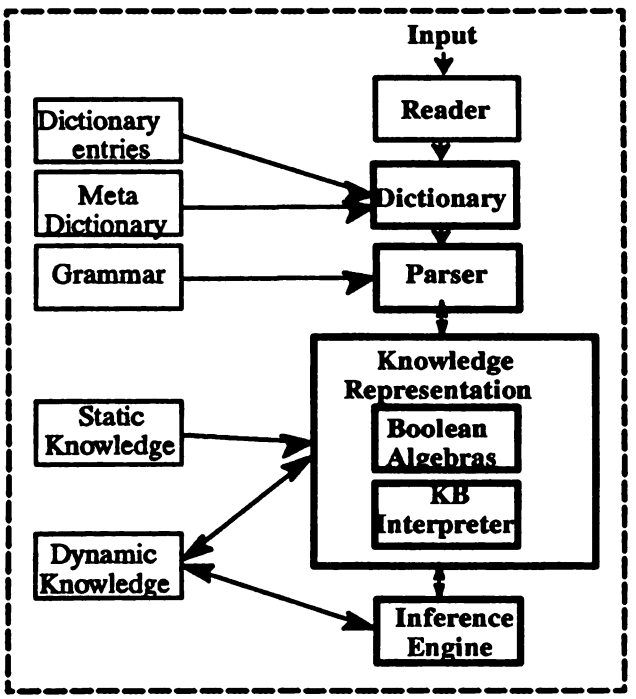


Figure 5: The Architecture of the UNO Natural Language Processing System

5 FUTURE WORK

The limitations of the model (see Section 2.5) constitute the directions of future work. The model provides insight into addressing some of these problems. Many modal operators and temporal expressions are scalars, and the UNO system can correctly compute entailments such as the entailment between *not believe* and *not know*, or between *never* and *not always*.

It is straightforward for the UNO system to be capable of scalar implicature [Horn, 1989], one type of non-logical, cancellable inference in natural language. This inference is a consequence of the incompleteness of people's knowledge and social norms of politeness, and is extremely common in natural language. Two examples of scalar implicature are: 1) the sentence *Some swans are white* which non-logically implicates *Not all swans are white*, but is logically consistent with *All swans are white*; and the sentence *He is not extremely bright* which can [6] convey the meaning *He is stupid*. I provide two out three elements necessary for handling scalar implicature: a computational model of the semantics of negation in natural language discussed in this paper, and a computational model of the semantics of scalar predicates discussed in [Iwańska, 1992]. The missing part is a theory of scalar implicature implemented as a function recognizing conditions that trigger and cancel scalar implicature.

6 Conclusion

I have presented a formal computational model of negation in natural language implemented as part of the UNO natural language processing system. The model allows the system to represent and reason with sentences with explicit negative information. I have demonstrated that Boolean properties of natural language, both in terms of representation and inference, can be successfully modeled with knowledge representation formalisms with Boolean semantics.

Acknowledgements

I thank David Wilkins, Nachum Dershowitz, Marianne Winslett, Dale Russell, Jeff Gordon, the members of the Knowledge-Based System Group in the Department of Computer Science of the University of Illinois at Urbana-Champaign, Paul Nielsen, and Mel Simmons for their comments. I also thank Erhard Hinrichs for his help at the initial stage of this research.

This research was done at the Knowledge-Based System Group, Department of Computer Science, University of Illinois at Urbana-Champaign, where it was supported in part by ONR grant N00014-88K0124, and at the Artificial Intelligence Laboratory of GE Research and Development Center.

[6] I refer to the meaning of this sentence when negation is not used metalinguistically.

References

[Aït-Kaci, 1984] Aït-Kaci, H. (1984). *A Lattice-Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures.* PhD thesis, Dept. of Computer and Info. Science, Univ. of Pennsylvania.

[Aït-Kaci and Nasr, 1985] Aït-Kaci, H. and Nasr, R. (1985). LOGIN: A logic programming language with built-in inheritance. MCC Technical Report Number AI-068-85, Microelectronics and Computer Technology Corporation.

[Barwise and Cooper, 1981] Barwise, J. and Cooper, R. (1981). Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4:159-219.

[Bobrow and Winograd, 1985] Bobrow, D. G. and Winograd, T. (1985). An overview of KRL, a knowledge representation language. In *Readings in Knowledge Representation*, pages 263–286. Morgan Kaufmann Publishers, Inc.

[Brachman et al., 1985] Brachman, R. J., Fikes, R. E., and Levesque, H. J. (1985). Krypton: A functional approach to knowledge representation. In *Readings in Knowledge Representation*, pages 411–430. Morgan Kaufmann Publishers, Inc.

[Clark, 1978] Clark, K. L. (1978). Negation as failure. In Gallaire, H. and Minker, J., editors, *Logic and data bases.* New York Plenum Press.

[Dawar and Vijayashanker, 1989] Dawar, A. and Vijayashanker, K. (1989). A three-valued interpretation of negation in feature structures. In *27-th Annual Meeting of the Association of Computational Linguistics.*

[Dowty et al., 1981] Dowty, D. R., Wall, R. E., and Peters, S. (1981). *Introduction to Montague semantics.* D. Reidel Publ. Co.

[Geffner, 1989] Geffner, H. (1989). Beyond negation as failure. In *Proceedings of the First International Conference on Knowledge Representation and Reasoning.*

[Hamm, 1989] Hamm, F. (1989). *Naturlich-sprachliche quantoren.* Max Niemeyer Verlag.

[Horn, 1989] Horn, L. R. (1989). *A Natural History of Negation.* The University of Chicago Press.

[Horty and Thomason, 1990] Horty, J. F. and Thomason, R. H. (1990). Boolean extensions of inheritance networks. In *Proceedings of 8-th National Conference on Artificial Intelligence*, pages 633–639.

[Iwańska, 1989] Iwańska, L. (1989). Automated processing of narratives written by 6-12 grade students: The BILING program. Technical Report UIUCDCS-R-89-1508, Dept. of Computer Science, University of Illinois at Urbana-Champaign.

[Iwańska, 1992] Iwańska, L. (1992). *A General Semantic Model of Negation in Natural Language: Representation and Inference.* PhD thesis, Dept. of Computer Science, University of Illinois at Urbana-Champaign.

[Johnson, 1990] Johnson, M. (1990). Expressing disjunctive and negative feature constraints with classical first-order logic. In *28-th Annual Meeting of the Association of Computational Linguistics.*

[Karttunen, 1984] Karttunen, L. (1984). Features and values. In *10-th International Conference on Computational Linguistics.*

[Kasper and Rounds, 1986] Kasper, R. T. and Rounds, W. C. (1986). A logical semantics for feature structures. In *24-th Annual Meeting of the Association of Computational Linguistics.*

[Keenan and Faltz, 1985] Keenan, E. L. and Faltz, L. M. (1985). *Boolean Algebra Semantics Of Natural Language.* D. Reidel Publ. Comp.

[Lifschitz, 1987] Lifschitz, V. (1987). On declarative semantics of logic programms with negation. In *Readings in Nonmonotonic Reasoning*, pages 337–350. Morgan Kaufmann Publishers, Inc.

[Moshier and Rounds, 1987] Moshier, M. D. and Rounds, W. C. (1987). A logic for partially specified data structures. In *14-th Symposium on Principles of Programming Languages.*

[Nakazawa et al., 1988] Nakazawa, T., Neher, L., and Hinrichs, E. W. (1988). Unification with disjunctive and negative values. In *ECAI-88.*

[Padgham, 1989] Padgham, L. (1989). *Non-Monotonic Inheritance for An Object-Oriented Knowledge-Base.* PhD thesis, Dept. of Computer and Info. Science, Linkoping Univ., Sweden.

[Pereira, 1987] Pereira, F. (1987). Grammars and logics of partial information. In *Proceedings of of the International Conference on Logic Programming*, volume 2, pages 989–1013.

[Reiter, 1985] Reiter, R. (1985). On reasoning by default. In *Readings in Knowledge Representation*, pages 402–410. Morgan Kaufmann Publishers, Inc.

[Schmolze, 1989] Schmolze, J. (1989). Terminological knowledge representation systems supporting n-ary terms. In *Proceedings of the First International Conference on Knowledge Representation and Reasoning.*

[Schwartz, 1989] Schwartz, D. G. (1989). Outline of a naive semantics for reasoning with qualitative linguistic information. In *IJCAI-89.*

[Sterling and Shapiro, 1986] Sterling, L. and Shapiro, E. (1986). *The Art of Prolog.* The MIT Press, Cambridge.

Conversational Events and Discourse State Change: A Preliminary Report

Massimo Poesio
Department of Computer Science
University of Rochester
Rochester, NY 14627-0226
poesio@cs.rochester.edu

Abstract

Event-based models of belief update like those proposed by Perrault, Appelt and Konolige are largely compatible with the proposals advanced in the literature on formal approaches to discourse interpretation, especially the recent work by Asher and Kamp, yet allow a close integration between the processes tracking the attentional state and performing intention recognition, and, once augmented with a more expressive representation for events and situations, afford a formalization of phenomena like discourse segmentation and focus shift. The logic presented in this paper, besides allowing reasoning about the way conversational events group in conversational threads, is also appropriate as a target of a formal translation procedure, so that a discourse interpretation procedure alternative to the DRS construction algorithm can be specified.

1 INTRODUCTION

Work on intention recognition by Allen, Cohen, Levesque, Perrault, and others [Cohen and Perrault, 1979; Allen and Perrault, 1980; Grosz and Sidner, 1986; Carberry, 1990] and especially recent work by Cohen and Levesque and Perrault [Cohen and Levesque, 1986; Perrault, 1990; Appelt and Konolige, 1988] led, in the words of Perrault ([Perrault, 1990], p. 162) to "a theory of speech acts in four parts:"

1. an account of propositional attitudes
2. a theory of action and its relation to attitudes
3. a description of the effects of locutionary acts on the mental state of the participants
4. definitions of the performance of illocutionary acts as the performance of [an] action ...by a speaker holding certain intentions, resulting in an update of the mental states of speaker and hearer.

In the literature on discourse understanding after Stalnaker, on the other hand, [Stalnaker, 1972; Lewis, 1979; Webber, 1978; Kamp, 1981; Heim, 1982; Reichman, 1985; Grosz and Sidner, 1986; Groenendijk and Stokhof, 1990] one finds widespread support for the hypothesis that the participants in a conversation share at any moment an *information state* which gets updated as the conversation goes along, and the thesis that '...the meaning of a sentence resides in its information change potential',[1] that is, that the meaning of a sentence is a function from information states to information states. The best known among these models, DRT, is being developed by Asher and Kamp [Asher, 1991; Kamp, 1990] into a model of *belief update*, in which information states are explicitly identified with mental states with a certain structure.

The two paradigms appear tantalizingly close, yet I am not aware of any work aimed at bridging the remaining gap. In this paper I present some preliminary work towards a model of discourse state change which is derived from the work on intention recognition yet incorporates the main insights of the account of belief developed by Asher and Kamp, mainly motivated by facts about reference. I claim that a model in which utterances are explicitly treated as actions does not simply afford a closer integration between the processes tracking the attentional state and performing intention recognition;crucially, it also gives the opportunity of formalizing a number of phenomena that at the moment DRT is not capable of handling, including the organization of utterances into discourse segments and the way the 'focus of attention' shifts during the conversation [Grosz, 1977; Linde, 1979]

The content of the paper is as follows: In the next section I present the data about reference and discourse interpretation that motivated this work. I then present the account of belief given by Kamp, and the work on belief update by Perrault, Appelt and Konolige. In order to obtain from the action-based paradigm a representation with the structural characteristics required by Kamp, and that can be used to accomplish the same task, I have taken as a

[1] This particular quotation is from [Groenendijk and Stokhof, 1990], but statements to this effect can be found in [Barwise and Perry, 1983; Devlin, 1991], and elsewhere.

starting point a situation-theoretical logic, *Episodic Logic* [Schubert and Hwang, 1990], which can be used as a translation for natural language utterances and which incorporates a notion of accessibility analogous to that of DRT. I present this logic in section 5, together with my augmentations (mostly constructs for incorporating an account of belief revision inspired by that of Appelt and Konolige). I then introduce the main proposal of the paper, that of introducing situation-like objects called *Discourse States* to satisfy Kamp's requirements about the representation of mental states. I briefly present the process by which discourse states get introduced and augmented in section 7, and in section 8 I show how the model allows the formalization of a number of discourse interpretation processes, including focus shift.[2]

2 THE DATA

The aim of the TRAINS project [Allen and Schubert, 1991] is to develop a natural language understanding system able to engage in conversations with a human user (the *manager*) whose task is to develop plans for transporting goods by train. The role of the system in these conversations is to assist the manager in developing the plan. An example of the kind of tasks the manager has to develop plans for is given in (1).

(1) I have to get one tanker of orange juice to Avon, and a boxcar of bananas to Corning, by 3pm.

An important characteristic of the TRAINS project is the decision of relying on transcripts of actual conversations in order to get an accurate picture of the pragmatic aspects of discourse understanding. At the moment, our corpus consists of twelve transcripts[3] of conversations between two speakers, in which one speaker plays the role of the system and the second speaker plays the role of the manager. The 'manager' and the 'system' are separated by a wall, and communicate via microphone; each has a copy of the map in Fig. 1.

The discussion in this paper will be concerned, for the most part, with the fragment of the transcript in (2). The manager's utterances are marked with 'M', the system's utterances with 'S'. The identification numbers of the utterances consist of two parts: the first number is the turn number, while the second indicates the utterance position within its turn.[4]

(2)
...
13.1 M: not at the same time
13.2 ok
13.3 We're gonna hook up engine E2 to the boxcar at Elmira,
13.4 and send that off to Corning
13.5 now while we're loading that boxcar with oranges at Corning,
13.6 we're gonna take the engine E3
13.7 and send it over to Corning,
13.8 hook it up to the tanker car,
13.9 and send it back to Elmira
14.1 S: okay
14.2 We could use one engine to take both the tanker and the boxcar to Elmira
15.1 M: oh,
15.2 we can do that?
16.1 S: yeah
...
29.1 M: okay,
29.2 great
29.3 while this is happening,
29.4 take engine E1 to Dansville,
29.5 pick up the boxcar,
29.6 and come back to Avon
30.1 S: okay
31.1 M: okay
31.2 then load the boxcar with bananas

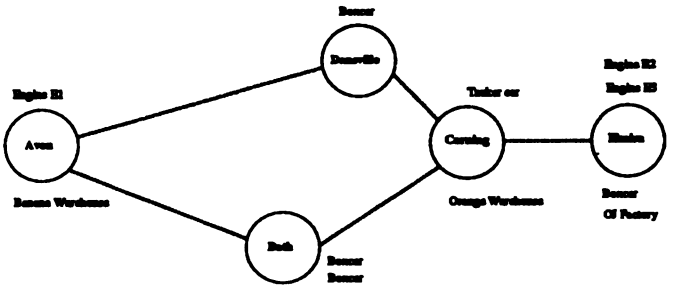


Figure 1: The map used by the participants in the conversation.

The definite descriptions "the boxcar" in 13.3 or "the tanker car" in 13.8 are instances of *immediate situation use* of definite NP's, which occurs when the object referred to is visible to both speaker and hearer in the situation of utterance, and is furthermore unique ([Hawkins, 1978], p.110). The plan discussed in (2) involves two boxcars, one in Elmira and one in Dansville. In 29.5 the focus of attention is apparently Dansville, since the reference to "the boxcar" is unambiguous even though three other boxcars are present in the map. Yet, Dansville clearly isn't the focus of attention during the whole dialogue, since another boxcar is discussed in 13.3-16.1, and at no moment in the discussion do the manager and the system seem to perceive an ambiguity, not even when "that boxcar" is used in 13.5.[5] In order to model the immediate situation use, we need to represent the fact that the speaker's attention is focused at certain times on some objects or situations, and that this *focus of attention* changes during a conversation [Grosz, 1977;

[2] In [Poesio, 1992] I discuss more in detail how the model presented here has been used to develop an account of attentional state change related to, although not identical with, the model proposed by Grosz and Sidner [1986]

[3] Collected by James Allen and Derek Gross.

[4] This notation is due to David Traum.

[5] The interpretation of "the tanker car" in 13.8 also seems to involve a reference to the current focus of attention, but this case is less clear-cut because only one tanker car exists in the domain.

Linde, 1979].

Grosz and Sidner [1986] proposed that the changes in attentional state are the result of "...a set of transition rules that specify the conditions for adding and deleting spaces (p. 179)." Because Grosz and Sidner's intention is to develop an *abstract* rather than a *processing* model of discourse structure (p. 176), they do not address questions like: What form are the transition rules rules going to take? Under which conditions are they going to be applied—when, for example, does an utterance trigger a transition rule? How do the transition rules relate to the rules for modifying the intentional and linguistic structures? Without an answer to these questions, however, it is hard to understand clearly what the theory predicts. A formal theory of discourse must provide an answer to these questions; yet, little has been attempted by way of a formal proposal for modeling focus shift.[6]

The fragment in (2) displays an instance of another well-known phenomenon, namely, the fact that when a definite description is used anaphorically,[7] the only antecedents considered are those in the same 'discourse segment' [Reichman, 1985; Grosz and Sidner, 1986; Fox, 1987]. Both "the boxcar" in `14.2` and "the boxcar" in `31.2` are cases of anaphoric use, yet "the boxcar" in `31.2` is unambiguous.

Finally, (2) illustrates the need for interaction between the processes tracking attentional state and those performing intention recognition as recognized early on by Hobbs [1979]. Consider `31.2`, for example. If the interpretation of the anaphoric definite "the boxcar" were to take place before intention recognition has been performed, the discourse segment which `31.2` is a part of would not have been determined yet, hence all potential referents ought to be considered. Conversely, if intention recognition were to take place before the referent for "the boxcar" has been identified, the plan reasoner ought to verify which action among all the actions involving boxcars in the plan is being discussed. That this interaction takes place is a rather compelling arguments aganst having the algorithm for belief update depend on syntactic information only.

3 BELIEF UPDATE IN DRT

In recent years, Asher, Kamp and others have been developing a theory of belief update based on the claim that some characteristics of mental states like belief suggest that belief states are structured in a way reminiscent of DRS's [Asher, 1991; Kamp, 1990]. What this attempt appears to be leading to is, in effect, an attempt of reformulating the DRT theory of anaphoric interpretation in terms of operations on belief states, thus offering a welcome opportunity to link the theory with theories concerned with other aspects of discourse reasoning.

[6] But see [Kamp, 1983; Pinkal, 1986].

[7] According to Hawkins, we have an anaphoric use when the definite article is used to refer to an object explicitly 'entered into the memory store' by a previous NP ([Hawkins, 1978], p.86).

It's not possible to provide more than a brief summary of either DRT or this recent work here; I'll refer the interested reader back to [Kamp, 1981; Heim, 1982; Kamp and Reyle, 1990] for an introduction to DRT and for most motivations. I'll follow Kamp's presentation rather than Asher's, merely because I'm more familiar with it. Kamp states rather clearly that much of his treatment is derived from Asher's work.

Kamp aims at developing "...a theory of belief which does justice to (i) the intentionality of beliefs and (ii) to the fact that their identity conditions are stricter than those of the corresponding sets of worlds." ([Kamp, 1990], p.30). He argues that the study of anaphoric elements can provide important clues about the likely form and organization of our attitudes; his key observation is that "the intended interpretation of such elements typically links them, and therewith the content of the sentence in which they occur, with the interpretation of the sentences preceding it." (p.34)

This would seem to indicate that some of the organizing principles of DRS are at play in the organization of beliefs as well. DRT is articulated around a system of rules—the so-called *construction algorithm*— which maps discourses belonging to the language into certain "interpretive structures". The output structures are called "Discourse Representation Structures" or "DRS's." The algorithm begins by adding the syntactic structure of the sentence to the current DRS. The rules are applied to the syntactic structure, and add *discourse referents* and *conditions* to the DRS. For example, the construction algorithm applied to sentence (3) results in the DRS in (4) ([Kamp, 1990], p.38):

(3) A whale beached near St.Diego.

(4)

x e p z

whale(x); event(e);
place(p); in(e,p); San Diego(z); near(p,z);
e:beached(x)

An important reason why Kamp feels DRS are an appropriate model of attitudes is their organization in discourse referents and conditions, and in particular, the fact that the discourse referents can act as links between conditions that originate in distinct sentences. A second reason why DRT appears to be a useful point of departure for a theory of attitudinal structure lies in its concern with the incremental character of language interpretation, and, therewith, of the assimilation of information by verbal means. Thus "...It is tempting to see DRT as providing a model of the process by which the recipient of a discourse acquires new beliefs as he takes in successive sentences."(p.40). Kamp thus proposes the following two principles about attitudinal states:

(A1) : Only the belief state as a whole determines well-defined truth conditions

(A2) : Complex attitudinal states must be assumed to be organized "around" the intentional elements they share.

These two principles provide the motivation for much of the development in the next sections.

A minimal requirement for the representation of attitudinal states is that it be able to distinguish between those components of the state that function as beliefs and those that function as desires. Kamp notes that we cannot model the cognitive state of an agent simply by assuming predicates of the form *Bel(Φ)*, for this would get us into an infinite regress: what is, in fact, *Bel(ϕ)*? A belief? Kamp argues that the fact that certain attitudes are beliefs (desires, etc.) is an inseparable part of the attitude themselves—their *mode*. He thus proposes to model attitudinal states in the following way:

(A3) : Complex attitudinal states have the structure of sets of pairs, where each pair consists of (i) a *mode indicator* (e.g., belief, desire) and (ii) a DRS. The different DRS's may share the same discourse referents even when they modes differ. Such sets of pairs are called *articulated DRS's*.

For example, utterance (3) would result in the following belief:

(5) ⟨*Bel*, | *x e p z* | whale(x); event(e); place(p); in(e,p); San Diego(z); near(p,z); e:beached(x) | ⟩

Kamp also introduces for each mode indicator M a corresponding two-place predicate M which takes as first argument discourse referents representing beings endowed with consciousness, and as second argument discourse referents representing propositions, so that *bel(A,P)* is also included in the language.

The theory of belief developed by Asher and Kamp includes a convincing account of a number of problems, among which the problem of the relation between beliefs and the outside world raised by *De Re* beliefs and the problem of belief sharing, that I won't have room to discuss here but are ultimately crucial for a theory of reference.[8] This makes it desirable to preserve the crucial insights of this theory. Kamp is deliberately vague on the axiomatization of belief he has in mind, but I won't be concerned with this issue here. Kamp himself points out what this theory needs the most: (i) an account of the logical force of attitudinal modes and a "higher order" part of the attitudinal state which consists of beliefs about how the agents came to acquire their beliefs, to allow for belief revision; and (ii) a better account of practical reasoning and how it integrates with the construction algorithm. This last point, of course, is related to the problems I observed in section 2— keeping track of the shifts in focus of attention and the relation between discourse segment recognition and reference.

As we will see in the next section, the work on action-based models of belief update concentrated on precisely these issues. This provides one motivation for attempting a marriage of the two theories. In addition, a model like Perrault's (that I will present in the next section) can be upgraded into a formalism which allows for talking about discourse segments and focus shifts, which are problems directly related to Kamp's main concern, reference interpretation.

[8] I discuss my use of (some of) these constructs in [Poesio, 1992].

4 ACTION-BASED MODELS OF BELIEF REVISION

The aim of the work on intention recognition of Allen, Cohen, Levesque, Perrault and others is to model the process by which a hearer gets to recognize the speaker's intentions in uttering a sentence. As the literal intention of an utterance can be rather different from the actual intention, recognizing these intentions may require complex reasoning. The representations of belief and action used in this literature have been developed to model this reasoning.

The unifying characteristic of these models is the assumption that by uttering a sentence, a speaker is performing a *speech act* [Austin, 1962; Searle, 1969]. A declared goal of some recent research, most conspicuously the work of Cohen and Levesque [Cohen and Levesque, 1990], is to derive the properties of speech acts observed in the earlier literature from general properties of actions, instead of stipulating illocutionary acts as primitives: "Utterance events are ...just a special case [of events] in which what is changed is the mental states of speakers and hearers." (From the introduction to [Cohen *et al.*, 1990], p.8.) Hence, part of the task of those engaged in this line of research is to develop a model of actions from which the general properties of conversations may be derived. For the purposes of this paper, the relevant aspect of the models of action proposed in this literature, is that the occurrence of an event results in certain *states* holding at the time after the event. This allows to write axioms specifying what the occurrence of a conversational event[9] tells about the intentions and/or beliefs of the speaker. For example, Cohen and Levesque's axiom for the use of declarative sentences states that if it is mutually known by speaker S and hearer H that e is an event of uttering sentence s to H in which S is the agent, that s is a declarative sentence with content p, and that H is attending S, then after the utterance it becomes the case that the H believes it is mutually believed that the speaker intends the hearer to recognize his intention that the hearer believes that the speaker believes that p is true:

(6) $MK_{S,H}$(Utter(H,s,e) & AGT(S,e) & Attend(H,S) & declarative(s,p))

[9] It is one of the assumptions of my work that linguistic items other than complete sentences —for example, discourse markers—may also result in changes in the discourse state. For this reason, I use in the paper the term conversational event to indicate those components of a discourse which result in a belief update. Speech acts are one type of conversational events.

$\supset$ after(e, $BMB_{H,S}G_SB_HG_SB_HB_Sp$)

(Where G_xp reads "x has the goal p" and B_xp reads "x has the belief that p.") Also resulting from a conversational event is a state of the hearer believing that a certain event occurred: Perrault calls this the *Observability* axiom ([Perrault, 1990], p.172):

Observability :
$\vdash DO_{x,t}\alpha \;\&\; DO_{y,t}Obs(x) \supset B_{y,t+1}DO_{x,t}\alpha$

In this formula, $DO_{x,t}\alpha$ reads "x did α at time t," while $B_{x,t}p$ reads "x believes at time t that p."

Perrault's main concern in [Perrault, 1990] is how one gets to infer, say, B_Hp from axioms for declarative sentences like (6). Perrault argues against Cohen and Levesque's formalization of the effect of conversational events in terms of monotonic axioms, on the grounds that it is impossible to find a weak enough belief that the illocutionary force of a declarative sentence can be characterized by H achieving that belief as a result of the sentence. Perrault proposes instead a *persistence theory of belief*, according to which the effects of speech acts are formulated as defaults in the sense of Reiter [Reiter, 1980], and in which it is assumed that old beliefs persist and that new ones are adopted as a result of observing external facts, provided that they do not conflict with old ones. Perrault assumes a weak S5 formalization of belief, to which Observability and the following two axioms are added:

Memory : $\vdash B_{x,t}p \supset B_{x,t+1}B_{x,t}p$

Persistence (P) : $\vdash B_{x,t+1}B_{x,t}p \supset B_{x,t+1}p$

Perrault also provides two default rules, one for declaratives and one for Belief Transfer ($\alpha \Rightarrow \beta$ stands for $\frac{\alpha:\beta}{\beta}$ in Reiter's notation):

Declarative : $DO_{x,t}(p.) \Rightarrow B_{x,t}p$

Belief Transfer : $B_{x,t}B_{y,t}p \Rightarrow B_{x,t}p$

Discussing the latter two rules or the choice of weak S5 as the basis for a formalization of belief would bring me too far beyond the scope of this paper. I will instead concentrate on whether Perrault's proposal gives us the right tools for correcting the problems with the DRT model of update that I mentioned in section 3.

Let us compare the two theories of belief update: if we ignore discourse referents and concentrate solely on adding conditions, we can characterize the DRT model in terms of Perrault's axioms as follows: in the model proposed by Kamp, (i) the rule for declaratives is a non-defeasible version of Perrault's default rule; (ii) there is no notion of time, hence no memory axiom about what happened 'in the past'; (iii) persistence is assumed and is, as in Perrault's case, non defeasible; and (iv) observability doesn't hold, that is, the occurrence of a conversational event is not included among the beliefs resulting from that conversational event.

The first advantage of Perrault's theory is that it allows for a much more intuitive treatment of declaratives. And as far as reference issues are concerned, I will note first of all that Perrault's model does allow for a natural formulation of focus shift rules along the lines sketched in the following axiom, whose intended interpretation is: If x utters α , and some other conditions occur, then the focus of attention will be shifted to β. (I will provide a more precise formulation of the idea in section 8.)

(7) $\vdash DO_{x,t}\alpha \;\&\; \Phi \supset B_{y,t+1}\,focus = \beta$

In addition, as we will see in section 6, including a version of Observability, together with an improved event ontology, allows for an account of discourse segmentation.

I believe these are good reasons for adopting an account of belief update like that proposed by Perrault. This theory, however, fares less well in terms of capturing Kamp's observations about the structural properties of belief, or for that matter as a theory of utterance representation. Perrault's logic is not really suited to serve as a translation language appropriate to deal with semantic phenomena like tense, quantifiers, etc. As far as the first problem is concerned, the key idea will be to introduce the notion of *situation* from Situation Theory: the result of the occurrence of a conversational event is not simply going to be inferring new beliefs holding after the conversational event, but also to organize these beliefs into a *discourse state* with certain constituents. Introducing situation theory will result in a natural solution for the discourse segments problem as well. In the next section I will present a version of Situation Theory, called Episodic Logic, which also includes a language that can be used as the target of a formal grammar, thus fixing the other problem with Perrault's formalism.

Before doing that, however, there is something more to be said about Persistence. I mentioned before that both DRT and Perrault's theory assume older beliefs to persist without revision. This feature of the theories is not desirable, because it results in belief revision never taking place— former beliefs always hold true, and block the acquisition of new beliefs through the rule for declaratives when this would generate in a conflict— and is in fact mentioned by Kamp as one of the problems with the DRT construction algorithm. However, Perrault's formalization of Persistence as an axiom, rather than as a default, is not a necessary feature of event-based models of belief update. It is instead forced upon him by his choice of Default Logic as his nonmonotonic framework; in Default Logic, treating Persistence as a default might create conflicts with the default rule for interpreting declaratives, thus originating multiple extensions.

Appelt and Konolige proposed a nonmonotonic logic called HAEL (Hierarchical AutoEpistemic Logic) to fix thix problem [Appelt and Konolige, 1988]. HAEL is meant to model an agent's belief state, and is based on the idea of imposing an ordering on the facts of a theory τ representing what the agent believes, thus inducing a hierarchy of subtheo-

ries $\tau_0 \prec \ldots \prec \tau_i \prec \ldots$. The 'lowest' subtheories include the facts believed with stronger evidence. The language of HAEL consists of a standard first-order language, augmented by an indexed set of unary modal operators L_i. The formula $L_i p$ is to be read as "p is an element of subtheory τ_i." This leads to restricting the use of L_i to the subtheories τ_j for $j \leq i$, as well as to the following conditions:

1. If $\phi \in T_j$, and $\tau_j \preceq \tau_i$, then $L_j\phi \in T_i$
2. If $\phi \notin T_j$, and $\tau_j \prec \tau_i$, then $\neg L_j\phi \in T_i$
3. If $\phi \in T_j$, and $\tau_j \prec \tau_i$, then $\phi \in T_i$

If we extend HAEL's language to include a class of modal self-belief operators indexed by time $L_{0,0}$, $\ldots L_{0,1}$ $\ldots\ldots L_{0,t}$ $\ldots L_{i,t}$ we can give a formulation of Persistence which avoids the problems with Perrault's:[10]

Persistence (HAEL) $L_{j,t+1}L_{j,t}\Phi \supset L_{j,t+1}\Phi$

According to this version of Persistence, self-beliefs 'maintain their strength' across time, so that older beliefs with strong evidence persist, whereas the weaker ones may be overridden by newest beliefs. The self-belief operators indexed by theory allow for a formalization of the fact noted by Kamp that attitudinal modes have different 'logical force'. This can be done by using, instead of a single attitude mode for belief *Bel*, an indexed set of them Bel_0 $\ldots Bel_i$. I will show in the next section how I propose to incorporate these ideas into Episodic Logic.

5 EPISODIC LOGIC

Due to the fact that it is being developed as the translation language for an English GPSG grammar [Schubert and Pelletier, 1982; Schubert and Hwang, 1990], Episodic Logic[11] is a fairly conservative version of Situation Theory[Barwise and Perry, 1983; Devlin, 1991], both in its syntax and in its semantics. The language of Episodic Logic is a conventional typed language with restricted quantification, abstraction and adverbial operators; on the semantic side, Episodic Logic is closer to the logic proposed by Kratzer [Kratzer, 1989] than to the mainstream version of Situation Theory as exemplified, say, in [Devlin, 1991].

The interesting aspect of Episodic Logic, as far as this paper is concerned, is the fact that its language allows reference to *situations*. Situations are collections of facts reflecting not so much how the world is organized, but rather how we organize our information about it. More specifically, the language of Episodic Logic includes operators for talking about *truth at a situation*. The * operator is similar to the $\models$ relation of more mainstream versions of Situation Theory. The fact `[f * s]`, to be read *f characterizes s*, is equivalent to $s \models f$ in the notation used by Devlin, and is true in a model iff the fact f is true in the situation denoted by s. In addition, Episodic Logic also includes a 'stronger' ** operator, which allows us to talk about *complete* characterizations of situations: The fact `[f ** s]` is true if and only if f is a complete characterization of s, that is, if for all expressions g such that $f \models g$, `[g ** s]` iff $f \equiv g$. The * and ** operators can be used to write axioms describing reasoning about situations; most of the axioms in the rest of the paper have in fact the form

$$[\Phi * s] \wedge \ldots [\Psi * s'] \supset [\Theta * s'']$$

All versions of Situation Theory include constructs for talking about *events*. The fact that events have temporal *locations* is described in Episodic Logic using the predicate `at-about`: `at-about(e,l)` reads "the event e has temporal location l." (For simplicity, I consider only temporal locations here.) The ** operator is used to characterize events: thus, the translation of "John left" in Episodic Logic is as in (8). This expression reads that e is an event of John leaving which takes place at location l.[12]

(8) `(∃ e at-about(e,l) ∧ before(l,now)`
`[leave(john) ** e])`

The set of logical operators of Episodic Logic that I will use also includes the kind-forming operator `K` [Schubert and Hwang, 1990], a set of temporal predicates which includes `before`, a causal predicate `cause`, a `subepisode-of` relation between situations analogous to the part-of $e_1 \subset e_2$ relation used by Barwise and Perry [Barwise and Perry, 1983], and a `constituent-of` predicate between objects and situations also analogous to that used by Barwise and Perry.

It is assumed in Episodic Logic, as in DRT, that certain NP's introduce discourse referents in the current discourse state. The approach taken in Episodic Logic is however inspired by the work in Dynamic Semantics [Barwise, 1987; Groenendijk and Stokhof, 1990; Rooth, 1987]: the semantic valuation procedure takes an additional parameter, called the *state*, and existentials and definite descriptions are treated as quantifiers, but they result in the addition of a new discourse referent to the state.

As far as the problem of belief revision is concerned, the focus of this paper is not to propose a new theory for choosing between extensions in belief revision, but to show that with the adoption of a more sophisticated ontology and some straightforward additions to the language we can obtain from the event-based models of belief revision a model of discourse interpretation compatible with the basic ideas of DRT and yet which allows for an intuitive treatment of several interesting discourse phenomena. Therefore, all I

[10]This axiom is mine.

[11]The term 'episode' was originally introduced to have a separate term for abstract situations. Currently, episode is synonymous with situation.

[12]In (8), as in the rest of the paper, 'e' variables like e or ce to denote events, and 's' variables like s or ds for statives. I also adopt the Episodic Logic convention of using square brackets for infix operators like **. Finally, I use throughout 'numbered variables' `ce31.2`, `e31.2` etc. when presenting the translation of utterance `31.2`

am going to do is to adapt the language of HAEL seen in the last section to my purposes, since HAEL seems to be the approach to formalizing belief revision most compatible with Kamp's proposal, as I understand it.[13] As suggested in the previous section, I am going to add to the language a family of modal self-belief operators Bel_0 ...Bel_i, analogous to the modal L_j operators of Appelt and Konolige, and playing the role of Kamp's *Bel* mode; and a correlated family of belief relations $believe_0$ ...$believe_i$, much as proposed Kamp. I will call the resulting language $\mathcal{EL}^H$. For example, Perrault's observability axiom can be rephrased as stating that a (conversational) event y of S telling H that Φ results in the hearer acquiring a belief x characterized by H $believing_1$ that S told H that Φ . The resulting belief is described in $\mathcal{EL}^H$ by the following formula:

```
(9)  (∃ x at-about(x,l)
       [believe1(H,
        ((∃ y at-about(y,l')
         [tell(S,H,Φ ) ** y])
        ** x]))
     ∧ cause(x,y)
```

In this paper I will only use $believe_0$ and $believe_1$.

I will be deliberately vague about the semantics of belief, since Kamp and Asher haven't yet presented a detailed proposal. As for the rest of Perrault's axioms, I'm going to adopt (the defeasible version of) Persistence and Memory, but I will discuss these in section 6 after introducing some additional concepts. I will ignore the problem of formalizing intentions.

6 DISCOURSE STATES AND CONVERSATIONAL EVENTS

This section will be dedicated to showing how the situation-theoretic framework presented in the previous section can be used to incorporate Kamp's postulates about belief (A1) and (A2) in an event-based representation. In the next section I will discuss how the construction procedure is defined.

As we said in the previous section, Perrault's observability axiom results in the belief described by (9). For example, utterance 31.2 in (2):

31.2 M: then load the boxcar with bananas.

translates into the following $\mathcal{EL}^H$ expression (I will discuss in section 7 how the translation is obtained), to be read: a belief of the system bs31.2 holds at temporal location l, of the system $believing_1$ that the manager instructed the system to load the boxcar with bananas.

```
(10) (∃ bs31.2 at-about(bs31.2,l)
       [believe1(sys,
        ((∃ ce31.2 at-about(ce31.2,l')
         [instruct(man,sys,
          (∃ e31.2 at-about(e31.2,l")
             [(the y boxcar(y) ∧
                   constituent-of(y,Ṡ)
              [sys
               ((with (K banana)) (load y))])
              ** e31.2]))
           ** ce31.2])
        ** bs31.2]))
      ∧ cause(ce31.2,bs31.2)
```

In Perrault's axiom, the persistence axiom ensures that the beliefs held by H prior to a conversational event still hold after it, unless those beliefs are revised. A form of this axiom analogous to Persistence(HAEL) could easily be formulated using $\mathcal{EL}^H$ (see section 4). However, as I mentioned before, this formulation fails to capture Kamp's claim that the mental state of the participants in a conversation does not consist of isolated attitudes, but of a coherent set of attitudes about the same objects. The language of situation theory instead does allow us to formulate the claim, as follows:

DIscourse State Principle (DISP) : At the beginning of the conversation, and then after each conversational event, a CP (re)organizes a subset of her beliefs/desires relevant to the conversation in a situation called *Discourse State*[14]. The Discourse State includes, minimally, the belief that the conversational event occurred.

This principle ensures that at any moment in the conversation the mental state of each participant includes a 'connected' structure playing a role similar to that of the root DRS in Kamp's proposal. Actually it does more, because it also takes into account a problem not considered by Kamp, namely, how the the thoughts relevant to the conversation are kept separate from the other beliefs.

The fact that a new discourse situation ds' 'results' from the hearer's incorporating the new belief into the set of 'relevant mental states' which hold at the temporal location l subsequent to a conversational event ce which took place when the discourse state was ds is represented in $\mathcal{EL}^H$ by the expression r(ds,ce) = ds'. The r operator allows the formulation of *Discourse State Change Axioms* (DSCA's), usually defeasible, formalizing the process by which the new discourse state gets defined. Perrault's persistence postulate is an especially important case of the kind of knowledge that is formulated in terms of discourse state change axioms; it translates as follows:

[13] An alternative would be using a probabilistic logic in which beliefs are augmented with probabilities. Schubert and Hwang are working on formalizing a version of Episodic Logic which includes a probabilistic version of material implication.

[14] I use the term Discourse State, rather than Discourse *Situation*, to avoid confusion with [Barwise and Perry, 1983; Devlin, 1991], in which the term is used to indicate what I call here conversational event.

Persistence:

```
[believe_i(A,Φ ) * s] ∧
[believe_1(A,r(s,e) = s') * s'] ⊃
[believe_i(A,Φ ) * s']
```

The following DSCA plays instead the role of Perrault's Memory axiom:

Memory:

```
[believe_i(A,Φ ) * s] ∧
[believe_1(A,r(s,e) = s') * s'] ⊃
[believe_0(A,[believe_i(A,Φ ) * s]) * s']
```

Other Discourse State Change Axioms specify how the focus of attention shifts as a result of a conversational event, and what kind of intentions are going to be part of the discourse state resulting from a certain conversational event. I show in section 8 how discourse state change axioms have been used to formalize a theory of focus shift and a proposal concerning intention recognition which involves using knowledge about the current state of the conversation.

According to the Discourse State Principle, the occurrence of 31.2 results in a belief which translates in the following $\mathcal{EL}^H$ expression:

```
(11) (∃ ds31.2 at-about(bs31.2,1)
      [believe_1(sys,
       [(∃ ce31.2 at-about(ce31.2,1')
        [instruct(man,sys,
         (∃ e31.2 at-about(e31.2,1")
            [(the y boxcar(y) ∧
                  constituent-of(y,Ṡ)
              [sys
               ((with (K banana))(load y))])
             ** e31.2]))
           ** ce31.2]))
       ∧ r(ds31.1,ce31.2) = ds31.2
      * ds31.2]))
```

Finally, let us consider the problem of modeling discourse segmentation. The organization of conversational events into discourse segments, I claim, is the result of a second situation-forming principle, according to which agents tend to group events into larger units. This phenomenon, evident in the process of understanding narratives [Nakhimovsky, 1988; Webber, 1988; Hwang and Schubert, 1992; Kameyama *et al.*, 1992], has been formalized in Situation Theory by adding a new object to the ontology, the *Course of events* [Barwise and Perry, 1983]. I use the predicate `subepisode-of(e,coe)` to assert that e is part of the course of events coe. I call the courses of conversational events *conversational threads.*

The effect of this principle on conversations can be formalized as follows: Grosz and Sidner proposed that whether a hearer inserts an utterance into a certain discourse segment depends on whether the intention(s) expressed by that utterance (the *discourse purpose*) are related to the intentions expressed by the discourse segment. Grosz and Sidner propose that intentions may be related in two different ways: when the discourse purpose is part of the satisfaction of another discourse purpose, the second purpose is said to *dominate* the first; if, instead, satisfying one intention is a prerequisite for satisfying a second one, the first intention is said to *satisfaction-precede* the second intention.

We may use the dominance and satisfaction-precedes relations between the discourse purposes expressed by conversational events to define derived relations among those conversational events, that I will call *dominance** and *satisfaction-precedes**. In this way we can formulate the following principle governing the process by which a hearer achieves the belief that a conversational event is part of a conversational thread:

CT Membership Principle (CTMP):

A conversational participant achieves the belief that a conversational event is part of a conversational thread iff the belief that that conversational event is dominated* or satisfaction-preceded* by another conversational event in that conversational thread is part of the discourse state of that participant.

The following $\mathcal{EL}^H$ expression translating the belief originated as the result of the occurrence of 31.2 revises (11) to incorporate the proposal that each event, including conversational events, is a subepisode of a course of events:

```
(12) (∃ ds31.2 at-about(bs31.2,1)
      [believe_1(sys,
       (∃ ce31.2 at-about(ce31.2,1') ∧
                 subepisode-of(ce31.2,coe1)
        [instruct(man,sys,
         (∃ e31.2 at-about(e31.2,1") ∧
             subepisode-of(e31.2,coe2)
            [(the y boxcar(y) ∧
                  constituent-of(y,Ṡ)
              [sys
               ((with (K banana)) (load y))])
             ** e31.2]))
           ** ce31.2])
       ∧ r(ds31.1,ce31.2) = ds31.2)
      * ds31.2]))
```

7 THE REVISED MODEL OF DISCOURSE INTERPRETATION

I propose a process of building the discourse state resulting from an utterance in three steps: first syntactic/semantic interpretation takes place, resulting in an $\mathcal{EL}^H$ expression called *Logical Form* (LF); then rules which play the role of the DRS construction rules produce expressions like (12); finally the rest of the pragmatic interpretation occurs.[15]

[15]There are reasons to believe that the process by which the interpretation of a sentence is added to the discourse state is actually *incremental*, in the sense that some pragmatic processing takes place before a complete interpretation for the sentence is

The Logical Form is generated by a GPSG grammar which uses $\mathcal{EL}^H$ as its target language but otherwise follows the principles proposed in [Schubert and Pelletier, 1982; Schubert and Hwang, 1990]. The Logical Form of `31.2` is the following $\mathcal{EL}^H$ expression:

```
(13) (decl
      (∃ e31.2 at-about(e31.2,l'') ∧
        subepisode-of(e31.2,Ċ)
       [(the y boxcar(y) ∧
            constituent-of(y,Ṡ)
         [sys ((with (K banana)) (load y))])
         ** e31.2]))
```

At this point in the interpretation neither the course of action of which `e31.2` is a part, nor the situation of interpretation of the boxcar have been determined, so that the LF at this point is only partially determined. (This is indicated by the use of the *parameters* $\dot{C}$ and $\dot{S}$.) The interpretation is going to be completed by the pragmatic processes, as discussed in [Poesio, 1992] and in section 8.

What kind of denotation to give to logical forms is still an open question. Not every type of utterance can be assigned a truth value as its denotation, and there is a fair amount of consensus that even in those cases in which this is possible (e.g., in the case of assertions), it is a much better strategy to assign an utterance a value which reflects its potential for context change [Heim, 1982; Barwise, 1987; Rooth, 1987; Groenendijk and Stokhof, 1990]. Different systems have been proposed, which we are currently considering. The simplest possibility would be to treat LF expressions as relations from situations to situations, as in the system proposed by Barwise and Rooth.

Each utterance is represented by a pair ⟨logical form,context⟩, in which the context includes the speaker and addresses of the utterance and its temporal location. This pair is input, together with the previous discourse state, to *Conversational Event Generation Rules* which finally produce the belief shown in (12). This system is thus closer to Kamp's than to Perrault's, in which presumably one would have rules producing an assertion of the form $DO_{x,t}\alpha$ after an utterance, from which assertion a belief would then be derived by means of the Observability axiom. The following Conversational Event Generation Rule is used to interpret declarative utterances:

```
(14) CE:⟨(decl (∃ e Φ (e) [Ψ ** e])),⟨S,H,T⟩⟩,
     Ds ⇝
     (∃ ds at-about(ds,l) ∧ T < l
      [believe1(H,
       (∃ ce at-about(ce,l') ∧ T = l' ∧
          subepisode-of(ce,Ċ)
        [tell(S,H,(∃ e Φ (e) [Ψ ** e]))
         ** ce ]))
       * ds ]
      ∧ r(Ds,e) = ds)
```

After a new discourse state is created, new beliefs and intentions may get added to it as the result of further pragmatic inferences, resulting for example from the process of interpreting definite descriptions or the process of intention recognition. Examples of these pragmatic processes will be given in the next section.

8 PRAGMATIC PROCESSING WITH DISCOURSE STATES

In this section I show how the model I have proposed can be used to formalize various types of discourse interpretation processes.

8.1 MODELING FOCUS SHIFT

Focus shift in our dialogues is apparently controlled by two principles. First of all, it appears that speaker and hearers perform a 'situation forming' operation not just to reorganize the discourse state after conversational events, but more in general whenever an event occurs—for example, to assess the state of the world after the action of sending engine E1 to Dansville, and this affects their use of reference. I am going to use the `r` operator to formalize this operation as well. Secondly, we observed the following principle:

Follow the movement: part of the intended effect of an utterance describing a movement action is to make the location 'at the end' of the movement the new 'position to look at' (i.e., the position described by *there*).

In the model of discourse change we have presented, this principle can be formulated as follows:[16]

```
[believe_i(y,r(ds,e) = ds') * ds'] ∧
[believe_i(y,
 [tell(x,y,intend(x,e':move(z,p))) ** e])
  * ds'] ∧
[believe_i(y,r(es,e') = es') * ds'] ⊃
  [believe_1(y,soa(x,y) = place(p,es')) * ds']
```

Where `place(p,e')` is the subepisode of `es'` (the situation 'resulting' from the action of moving `z` to `p`) consisting of the facts true at `p`, and `soa(x,y)` denotes the current *situation of attention*, if any (which plays a role similar to Barwise and Perry's 'object we are attending to' ([1983], pag. 87)).

obtained. There is evidence, for example, that referential expressions affect the parsing process; furthermore, our transcripts contain sequences in which multiple utterances by both speakers occur, each with clear pragmatic effects, before a complete interpretation is obtained. We are working on a revised version of the model presented here that takes these facts into account.

[16] All the unbound variables are to be taken as universally quantified.

Let us see now how these principles work in interpreting utterances 29.4 -29.5 in (2). The CE generation rules produce the following translation for 29.4:

```
(15) (∃ ds29.4 at-about(bs29.4,1)
      [believe1(sys,
       (∃ ce29.4 at-about(ce29.4,1') ∧
                 subepisode-of(ce29.4,coe1)
        [instruct(man,sys,
         (∃ e29.4 at-about(e29.4,1'') ∧
              subepisode-of(e29.4,coe2)
            [sys ((to Dansville)
                  (take E1))]
             ** e29.4]))
            ** ce29.4])
       ∧ r(ds29.3,ce29.4) = ds29.4)
      * ds29.4]))
```

Because of the 'follow the movement' principle, the object of attention shifts to Dansville in the discourse state resulting from the conversational event generated by the utterance of 29.4; i.e., the following fact is inferred:

```
[believe1(sys,
 [soa(man,sys) = place(dansville,es)])
 * ds29.4]
```

where es is the state of the world resulting from the occurrence of the event of moving the boxcar to Dansville, and place(dansville,es) is the subepisode of es consisting of the facts true at the spatial location Dansville in es.

Consider now the belief resulting from 29.5:

```
(16) (∃ ds29.5 at-about(bs29.5,1)
      [believe1(sys,
       (∃ ce29.5 at-about(ce29.5,1') ∧
                 subepisode-of(ce29.5,coe1)
        [instruct(man,sys,
         (∃ e29.5 at-about(e29.5,1'') ∧
              subepisode-of(e29.5,coe2)
            [(the y boxcar(y) ∧
                 constituent-of(y,Ṡ)
             [pickup(sys,y) ** e29.5]))
            ** ce29.5])
       ∧ r(ds29.4,ce29.5) = ds29.5)
      * ds29.5]))
```

The possibility of interpreting the definite "the boxcar" here as referring to the boxcar in Dansville (let us call that b1) is licensed by a (defeasible) principle for anchoring identifying situations called **PAIS1** in [Poesio, 1992], which formalizes the immediate situation use of definite descriptions. **PAIS1** says that if a speaker uses a referring expression "the P," and if the attention of the CP's is currently focused on the situation f, then it is plausible that f be the identifying situation for "the P." This is formalized by the following defeasible axiom schema:

(PAIS1)

```
[believei(y,
 [tell(x,y,(the z P(z) ∧
                constituent-of(z,Ṡ)
                Q(z))) ** e]) * s] ∧
[believei(y,r(s',e) = s) * s] ∧
[believei(y,soa(x,y) = f) * s] ⊃
[believe1(y,anchor(Ṡ,f)) * s]
```

(The statement anchor(Ṡ,f) 'fixes' the interpretation of the parameter Ṡ to f.) **PAIS1** allows the hearer to draw the inference that the identifying situation for z is place(dansville,es):

```
[believe1(y,
 anchor(Ṡ,place(dansville,es))) * ds29.5]
```

since the object denoted by the variable y is a constituent of place(dansville,es), and there is only one boxcar in that situation, the hearer can also infer that

```
[believe1(sys,y = b1) * ds29.5]
```

8.2 INTENTION RECOGNITION

The process of *recognizing* the speakers' intentions and how they relate to each other is very complex. Discussing the process of intention recognition in detail would require a separate paper. I would simply like to show here that the approach presented above allows for a relatively straightforward formalization of a proposal about intention recognition presented in [Poesio, 1991].

Conversational Analysts [Sacks *et al.*, 1974; Levinson, 1983] have suggested that the conversational participants have extensive knowledge both about the 'global structure' of specific types of conversations (in our case, plan elaboration conversations) and about their 'micro structure,' structured around *Adjacency Pairs* like Question-Answer or Inform-Acknowledge. This knowledge tells them about the preferred continuations, given the current discourse state, and plays an important role in recognizing the conversational events occurring in a conversation. This idea is equivalent to stipulating that the conversational participants know about 'default recipes for action', that I called *Discourse Scripts*.[17] At every moment, the CP's know that the current discourse state represents a certain position in the currently active discourse script; this tells them what the 'preferred next move' will be. A discourse script for plan conversations might look like this:

```
PLAN-CONVERSATION[man,sys]
  INTRODUCE TASK[man]
  DISCUSS RECIPE[man,sys]
  APPROVE RECIPE[man,sys]
  CLOSING[man,sys]
```

Only very few additions to the language I have been using are needed in order to use it to describe the knowledge

[17] The name is probably due to David Traum.

encoded in d-scripts. The 'current position' in a d-script is represented by including into the discourse state assertions of the form `state(CT) = N`, where `CT` is a conversational thread, and `N` one of the positions within a conversational thread. The notion of 'current' conversational thread is also useful: this can be encoded by facts of the form `current-ct = ct1`.

With these two additions, we can code d-scripts as defeasible $\mathcal{EL}^H$ axioms predicting the intention a CP expects to see fulfilled by the next utterance, unless a different intention is explicitly signalled. For example, the following defeasible axiom formalizes the expectations of the system at the beginning of the conversation, when the manager is supposed to introduce the task. The manager expects the system to reply with an acknowledgment to her report; this originates a new conversational thread subordinate to the plan conversation.

```
(17) [believe_i(sys,r(ds,ce) = ds') * ds'] ∧
     [believe_i(sys,
      [tell(man,sys,Φ ) ** ue]) * ds'] ∧
     [believe_i(sys,
      subepisode-of(ce,ct)) * ds] ∧
     [believe_i(sys,state(ct) = 0) * ds] ∧
     [believe_i(sys,plan-conversation(ct)) * ds]
     ⊃
        [believe_1(sys,
         (∃ ie at-about(ie,l)
          [intend(man,
                  describe-task-to(man,sys,Φ ))
                  ** ie]))
          * ds'] ∧
         [believe_1(sys,
          (∃ ct' introduce-task-AP(ct') ∧
                at-about(ct',l')
           dominates*(ct,ct') ∧
           state(ct') = 0))
          * ds'] ∧
         [believe_1(sys,
                   current-ct = ct) * ds']
```

9 CONCLUSIONS

I have shown how the event-based approach to belief revision can be used to solve some problems with the approach to belief revision based on DRT, and how to solve some representational problems related to reference phenomena by adopting the richer ontology of events and situations proposed in representation Situation Theory. The resulting model allows a formalization of a number of discourse interpretation processes.

At this stage I was mainly concerned with developing a representation that would enable me to deal with our transcripts; thus the formalization of the logic is not yet complete. The two main concerns in this direction are the precise characteristics of the theory of belief and the behavior of the theory of plausible reasoning I have adopted.

A detailed analysis of one of our transcripts, with all the inferential steps, is almost complete, and will be included in a future TRAINS Technical Note. The new implementation of the reference module of TRAINS incorporating these ideas is also under way.

Acknowledgements

I wish to thank my advisor Len Schubert and James Allen, George Ferguson, Janet Hitzeman, Megumi Kameyama, Rebecca Passonneau and David Traum for useful discussions. This work was supported by the Air Force-Rome Air Development Center Research contract no.F30602-91-C-0010.

References

[Allen and Perrault, 1980] Allen, J. and Perrault, C. 1980. Analyzing intention in utterances. *Artificial Intelligence* 15(3):143–178.

[Allen and Schubert, 1991] Allen, J. F. and Schubert, L.K. 1991. The TRAINS project. TRAINS Technical Note 91-1, University of Rochester, Department of Computer Science.

[Appelt and Konolige, 1988] Appelt, D. and Konolige, K. 1988. A practical nonmonotonic theory of reasoning about speech acts. In *Proc. ACL-88*, Buffalo.

[Asher, 1991] Asher, N. 1991. Reference to abstract objects in English. Unpublished manuscript.

[Austin, 1962] Austin, J. L. 1962. *How to Do Things with Words*. Harvard University Press, Cambridge, MA.

[Barwise and Perry, 1983] Barwise, J. and Perry, J. 1983. *Situations and Attitudes*. The MIT Press.

[Barwise, 1987] Barwise, J. 1987. Noun phrases, generalized quantifiers and anaphora. In Gärdenfors [1987]. 1–30.

[Carberry, 1990] Carberry, S. 1990. *Plan Recognition in Natural Language Dialogue*. The MIT Press, Cambridge, MA.

[Cohen and Levesque, 1986] Cohen, P. R. and Levesque, H. J. 1986. Persistence, intention and commitment. In Georgeff, M. and Lansky, A., editors 1986, *Reasoning About Knowledge and Action*. 297–340.

[Cohen and Levesque, 1990] Cohen, P. R. and Levesque, H. J. 1990. Rational interaction as the basis for communication. In Cohen, P.R.; J.Morgan, ; and Pollack, M., editors 1990, *Intentions in Communication*. Morgan Kaufmann. chapter 12, 221–256.

[Cohen and Perrault, 1979] Cohen, P. R. and Perrault, C. R. 1979. Elements of a plan based theory of speech acts. *Cognitive Science* 3(3):177–212.

[Cohen *et al.*, 1990] Cohen, P. R.; Morgan, J.; and Pollack, M., editors 1990. *Intentions in Communication*. MIT Press, Cambridge, MA.

[Devlin, 1991] Devlin, K. 1991. *Logic and Information*. Cambridge University Press, Cambridge, UK.

[Fox, 1987] Fox, B. A. 1987. *Discourse Structure and Anaphora*. Cambridge University Press, Cambridge, UK.

[Gärdenfors, 1987] Gärdenfors, P., editor 1987. *Generalized Quantifiers*. D. Reidel, Dordrecht, The Netherlands.

[Groenendijk and Stokhof, 1990] Groenendijk, J.A.G. and Stokhof, M.J.B. 1990. Dynamic Montague Grammar. In Groenendijk, J.A.G.; Stokhof, M.J.B.; Chierchia, G.; and Dekker, P., editors 1990, *Quantification and Anaphora I*. DYANA Deliverable R2.2.A.

[Grosz and Sidner, 1986] Grosz, B. and Sidner, C. 1986. Attention, intention, and the structure of discourse. *Computational Linguistics* 12(3):175–204.

[Grosz, 1977] Grosz, B. 1977. *The Representation and Use of Focus in Dialogue Understanding*. Ph.D. Dissertation, Stanford University.

[Hawkins, 1978] Hawkins, J. A. 1978. *Definiteness and Indefiniteness*. Croom Helm, London.

[Heim, 1982] Heim, I. 1982. *The Semantics of Definite and Indefinite Noun Phrases*. Ph.D. Dissertation, University of Massachusetts at Amherst.

[Hobbs, 1979] Hobbs, J. 1979. Coherence and coreference. *Cognitive Science* 3:67–90.

[Hwang and Schubert, 1992] Hwang, C. H. and Schubert, L. K. 1992. Tense trees as the "fine structure" of discourse. In *Proc. ACL-92*, Newark, DE. 232–240.

[Kameyama *et al.*, 1992] Kameyama, M.; Passonneau, B.; and Poesio, M. 1992. Temporal centering. In Preparation.

[Kamp and Reyle, 1990] Kamp, H. and Reyle, U. 1990. From discourse to logic. To appear.

[Kamp, 1981] Kamp, H. 1981. A theory of truth and semantic representation. In Groenendijk, J.; Janssen, T.; and Stokhof, M., editors 1981, *Formal Methods in the Study of Language*. Mathematical Centre, Amsterdam.

[Kamp, 1983] Kamp, H. 1983. SID without time or questions. Unpublished chapter of *Situations in Discourse*.

[Kamp, 1990] Kamp, H. 1990. Prolegomena to a structural account of belief and other attitudes. In Anderson, C. A. and Owens, J., editors 1990, *Propositional Attitudes—The Role of Content in Logic, Language, and Mind*. University of Chicago Press and CSLI, Stanford. chapter 2, 27–90.

[Kratzer, 1989] Kratzer, A. 1989. An investigation of the lumps of thought. *Linguistics and Philosophy* 12:607–653.

[Levinson, 1983] Levinson, S. 1983. *Pragmatics*. Cambridge University Press.

[Lewis, 1979] Lewis, D. 1979. Scorekeeping in a language game. *Journal of Philosophical Logic* 8:339–359.

[Linde, 1979] Linde, C. 1979. Focus of attention and the choice of pronouns in discourse. In Givon, T., editor 1979, *Syntax and Semantics 12*. Academic Press.

[Nakhimovsky, 1988] Nakhimovsky, A. 1988. Aspect, aspectual class, and the temporal structure of narratives. *Computational Linguistics* 14(2):29–43.

[Perrault, 1990] Perrault, C. R. 1990. An application of default logic to speech act theory. In Cohen, P. R.; Morgan, J.; and Pollack, M. E., editors 1990, *Intentions in Communication*. The MIT Press, Cambridge, MA. chapter 9, 161–185.

[Pinkal, 1986] Pinkal, M. 1986. Definite noun phrases and the semantics of discourse. In *Proc. COLING-86*, Bonn. 368–373.

[Poesio, 1991] Poesio, M. 1991. Expectation-based recognition of discourse segmentation. In *Proc. AAAI Fall Symposium on Discourse Structure*, Asilomar, CA.

[Poesio, 1992] Poesio, M. 1992. A situation-theoretic formulation of definite descriptions interpretation and focus shift in plan elaboration dialogues. Submitted for publication.

[Reichman, 1985] Reichman, R. 1985. *Getting Computers to Talk Like You and Me*. The MIT Press.

[Reiter, 1980] Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13(1–2):81–132.

[Rooth, 1987] Rooth, M. 1987. Noun Phrase Interpretation in Montague Grammar, File Change Semantics, and Situation Semantics. In Gärdenfors [1987]. 237–268.

[Sacks *et al.*, 1974] Sacks, H.; Schegloff, E.; and Jefferson, G. 1974. A simplest systematics for the organization of turn-taking for conversation. *Language* 50:696–735.

[Schiffrin, 1987] Schiffrin, D. 1987. *Discourse Markers*. Cambridge University Press, Cambridge.

[Schubert and Hwang, 1990] Schubert, L. K. and Hwang, C. H. 1990. An episodic knowledge representation for narrative texts. Technical Report 345, University of Rochester, Rochester, NY.

[Schubert and Pelletier, 1982] Schubert, L. K. and Pelletier, F. J. 1982. From English to Logic: Context-free computation of 'conventional' logical translations. *American Journal of Computational Linguistics* 10:165–176.

[Searle, 1969] Searle, J. 1969. *Speech Acts*. Cambridge University Press, New York.

[Stalnaker, 1972] Stalnaker, R. 1972. Pragmatics. In Davidson, D. and Harman, G., editors 1972, *Semantics of Natural Language*. D. Reidel Pub. Co., Dordrecht. 380–397.

[Webber, 1978] Webber, B. L. 1978. A formal approach to discourse anaphora. Report 3761, BBN, Cambridge, MA.

[Webber, 1988] Webber, B. L. 1988. Tense as discourse anaphor. *Computational Linguistics* 14(2):61–73.

VI.

Deduction

Learning Useful Horn Approximations

Russell Greiner*
755 College Road East
Siemens Corporate Research
Princeton, NJ 08540
greiner@learning.siemens.com

Dale Schuurmans
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4
dale@cs.toronto.edu

Abstract

While the task of answering queries from an arbitrary propositional theory is intractable in general, it can typically be performed efficiently if the theory is Horn. This suggests that it may be more efficient to answer queries using a "Horn approximation"; *i.e.*, a horn theory that is semantically similar to the original theory. The utility of any such approximation depends on how often it produces answers to the queries that the system actually encounters; we therefore seek an approximation whose expected "coverage" is maximal. Unfortunately, there are several obstacles to achieving this goal in practice: *(i)* The optimal approximation depends on the query distribution, which is typically not known *a priori*; *(ii)* identifying the optimal approximation is intractable, even given the query distribution; and *(iii)* the optimal approximation might be too large to guarantee tractable inference. This paper presents an approach that overcomes (or side-steps) each of these obstacles. We define a learning process, ADCOMP, that uses observed queries to estimate the query distribution "online", and then uses these estimates to hill-climb, efficiently, in the space of size-bounded Horn approximations, until reaching one that is, with provably high probability, effectively at a local optimum.

1 Introduction

Many performance systems compute answers to queries based on the information present in a knowledge base. Unfortunately, this can involve reasoning from an arbitrary propositional theory, which is inherently intractable (assuming $P \neq NP$) [Coo71, GJ79]. We describe a technique that "approximates" an arbitrary theory, transforming it into a representation that admits more efficient, if less categorical, reasoning [EBBK89]. In particular, our work extends the "knowledge compilation" method of Selman and Kautz [SK91]: Given a general propositional theory Σ, their compiler will compute a pair of "bracketing" Horn theories S and W, with the property that $S \models \Sigma \models W$.[1] Figure 1 shows how the resulting "compiled system" $PS = PS^{\alpha}(S, W, \Sigma)$ uses these bracketing theories to determine whether a query σ follows from Σ. If $W \models \sigma$, PS returns "yes"; otherwise, if $S \not\models \sigma$, then PS returns "no". Notice that these are the correct answers; *i.e.*, $W \models \sigma$ guarantees that $\Sigma \models \sigma$, and $S \not\models \sigma$ guarantees that $\Sigma \not\models \sigma$. Moreover, these tests are efficient; in fact, linear in the size of S, W and σ [DG84], provided $\neg\sigma$ is Horn.[2]

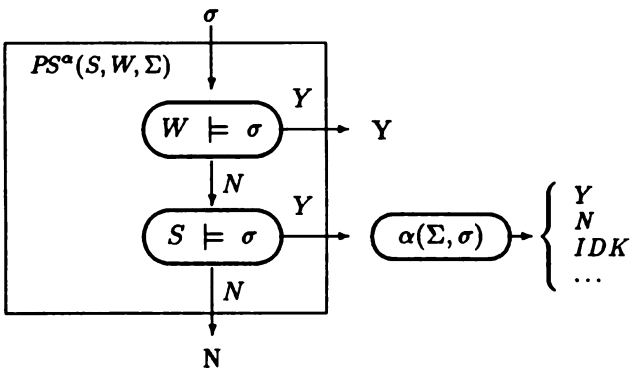


Figure 1: Flow Diagram of $PS^{\alpha}(S, W, \Sigma)$ addressing $\Sigma \models^{?} \sigma$

This paper extends [SK91]'s interesting results by ad-

*Some of this work was performed at the University of Toronto, where it was supported by the Institute for Robotics and Intelligent Systems, and by an operating grant from the National Science and Engineering Research Council of Canada. Both authors thank Bart Selman, Alon Levy, Radford Neal, Sheila McIlraith, Narendra Gupta and the anonymous referees for providing many helpful comments on this paper.

[1] We call each such S a "Strengthening" of the initial theory, and each such W an "Weakening". We assume each general theory is in clausal form — *i.e.*, expressed as a set (conjunction) of clauses, where each clause is a set (disjunction) of atomic literals, each either positive or negative. A theory is Horn if each clause includes at most one positive literal.

[2] We can actually allow the query σ to be a conjunction of "Horn-dual" propositions (CHD), where a proposition σ is a Horn-dual iff its negation $\neg\sigma$ is horn. Notice CHD strictly includes CNF.

dressing several unresolved issues.

ISSUE 1: WHICH α? It is not clear what PS should do if $W \not\models \sigma$ and $S \models \sigma$. For instance, we can consider various different classes of performance systems, each identified by its superscript: In particular, a PS^{IDK} system will simply return IDK (for "I don't know") in this situation, PS^{GUE} will "guess" at an answer, while the sound PS^{SND} will spend as long as necessary to compute whether $\Sigma \models \sigma$.

Of course, we want this problematic situation to occur rarely; we therefore prefer S and W theories that cover a maximal number of queries, as this means a minimal number of queries will fall through to the final α stage. [SK91] suggests restricting S (resp., W) to be a "weakest Strengthening", s_w (resp., a "strongest Weakening", w_s), which are the obvious extrema:

$$s_w(\Sigma, S) \iff \forall T\ [S \models T \models \Sigma\ \&\ \text{Horn}(T)] \Rightarrow S \equiv T$$

$$w_s(\Sigma, W) \iff \forall T\ [\Sigma \models T \models W\ \&\ \text{Horn}(T)] \Rightarrow W \equiv T$$

That article argues that such extrema are appropriate, as they cover a maximal number of queries. (To illustrate this idea, let W be a weakening that is not the strongest one — *i.e.*, $\Sigma \models w_s \models W$ where $W \not\models w_s$. Then there are queries σ such that $PS(S, w_s, \Sigma)$ would return Yes quickly, but $PS(S, W, \Sigma)$ will fall through to the problematic $\alpha(\Sigma, \sigma)$ step.)

There are, however, several complications associated with these extrema.

ISSUE 2: INTRACTABLE COMPILATION. The task of finding either extremum is intractable [SK91, p906], meaning they cannot be found efficiently (if $P \neq NP$).

ISSUE 3: MULTIPLE STRENGTHENINGS. There can be several weakest strengthenings. (For example, $\{a\}$ and $\{b\}$ each qualify as a weakest strengthening for $a \vee b$; *i.e.*, each satisfy $s_w(\{a \vee b\}, \cdot)$.)

ISSUE 4: EXPONENTIALLY LARGE WEAKENING. The cost of the first step of the $PS(S, W, \Sigma)$ process — *viz.*, determining whether $W \models^? \sigma$ — is linear in the size of W; unfortunately (the unique) strongest weakening w_s can be *exponential* in the size of the initial Σ. This means the resulting $PS^{\alpha}(S, w_s, \Sigma)$ system can still be intractable (even if we use a trivial $\alpha = IDK$, that simply returns IDK), as its first step can require exponential time.[3]

The rest of the paper presents an algorithm, ADCOMP, that addresses (and/or explicitly side-steps) each of these concerns. Section 2 describes this algorithm and shows how it deals with most of the issues; Section 3 then discusses several extensions to cope with the remaining points. The proof that ADCOMP works correctly appears in Appendix A.

[3] Notice that we encounter different problems when seeking optimal weakenings and strengthenings: There is a unique optimal weakening, but its size can be exponentially larger than $|\Sigma|$. By contrast, there can be many different optimal strengthenings; however each is essentially the same size as Σ; Subsection 2.4.

2 The ADCOMP Algorithm

The basic idea underlying our approach is to learn a reasonably-sized approximation that is likely to be good enough for the anticipated queries. Subsection 2.1 first motivates this approach; the rest of the section describes the ADCOMP algorithm ("ADaptive COMPiler") that implements these ideas. Subsection 2.2 states the fundamental theorem that specifies ADCOMP's functionality (whose proof appears in Appendix A). Subsection 2.3 provides the statistical foundations to motivate why this algorithm is feasible. Subsections 2.4 and 2.5 then present further details of the structure of the ADCOMP algorithm; and Subsection 2.6 discusses the algorithm's computational efficiency.

2.1 Our Approach

Tractable Inference. Given our objective of finding a representation of the given theory that admits efficient reasoning, we will (for now) consider only *polynomial-sized weakenings* (as this guarantees that $W \models \sigma$ can be answered efficiently) and to $PS^{IDK}(S, W, \Sigma)$ systems (as they are guaranteed to run efficiently, simply terminating with IDK whenever $W \not\models \sigma$ and $S \models \sigma$). These restrictions avoid the problems mentioned ISSUE 1 (WHICH α?) and ISSUE 4 (EXPONENTIALLY LARGE WEAKENING); Extension 6 in Section 3 will later return to these issues.

To state this more precisely: Given any propositional theory Σ, define $\text{Approx}_K(\Sigma)$ to be to the set of all Horn approximations of Σ whose sizes are at most K, *i.e.*,

$$\langle S, W\rangle \in \text{Approx}_K(\Sigma) \iff S \models \Sigma \models W\ \&\ \text{Horn}(W)\ \&\ \text{Horn}(S)\ \&\ |S| \leq K\ \&\ |W| \leq K$$

where the size of a horn theory $|T|$ is the number of clauses in T.[4] We identify each such Horn approximation $\langle S, W\rangle \in \text{Approx}_K(\Sigma)$ with the associated performance system $PS^{IDK}(S, W, \Sigma)$.

Utility of Horn Approximations. ISSUE 3 (MULTIPLE STRENGTHENINGS) noted there are many possible weakest strengthenings of a given theory; there are also many different K-sized strongest weakenings. How can we decide which to use? We adopt a pragmatic position: the optimal system is the one that has the *best expected performance* over the natural distribution of queries, based on a scoring function. For now, we define the scoring function to be simply the approximation's *coverage*.[5] Given any approximation $\langle S, W\rangle$ and query σ, let $c(\langle S, W\rangle, \sigma) \stackrel{def}{=} d(W, \sigma) + (1 - d(S, \sigma))$ where

$$d(T, \sigma) \stackrel{def}{=} \begin{cases} 1 & \text{if } T \models \sigma \\ 0 & \text{otherwise} \end{cases}.$$

[4] As the number of literals in each clause is at most L (where L = total number of variables), this measure is within a constant factor of the other obvious ways of measuring the size of a theory.

[5] Extension 6 in Subsection 3 considers other scoring functions. Notice higher scores are preferable.

Hence, $c(\langle S, W\rangle, \sigma) = 1$ iff σ is "covered" by $\langle S, W\rangle$, in that either $W \models \sigma$ or $S \not\models \sigma$.

This cost function evaluates $\langle S, W\rangle$'s performance for a single query. Our approximations, however, will have to solve an entire ensemble of problems; we clearly prefer the approximation that is best overall. We therefore define the "utility of the approximation $\langle S, W\rangle$" to be the expected value of this $c(\langle S, W\rangle, \cdot)$ scoring function, over the natural distribution of queries.

To state this more precisely: Given $\mathcal{Q}$, the set of all possible (CHD) queries, let $P: \mathcal{Q} \mapsto [0, 1]$ be the stationary distribution of the queries, where $P(q)$ is the probability of encountering the query $q \in \mathcal{Q}$. Then the utility measure used to evaluate an approximation $\Upsilon = \langle S, W\rangle$ is its expected score with respect to P,

$$\mathrm{C}[\Upsilon] \stackrel{def}{=} E[c(\Upsilon, \sigma)] = \sum_{\sigma \in \mathcal{Q}} P(\sigma) \times c(\Upsilon, \sigma) .$$

Our basic goal is to identify an "optimal K-sized approximation", which is an approximation $\Upsilon_{opt} \in \mathrm{Approx}_K(\Sigma)$ whose expected score is maximal:

$$\Upsilon_{opt} \in \mathrm{Approx}_K(\Sigma) \;\; \& \;\; \forall \Upsilon \in \mathrm{Approx}_K(\Sigma) \;\; \mathrm{C}[\Upsilon_{opt}] \geq \mathrm{C}[\Upsilon]$$

Hence, we are seeking a (reasonably-sized) horn approximation that is good *for the given distribution of queries*. Unfortunately, there are two major obstacles to achieving this goal; these are described in the next two points, and then addressed in the next subsections.

Learning. First, under the realistic assumption that the distribution P is unknown, there is no *a priori* way of determining the values of $\mathrm{C}[\cdot]$, and hence of determining which Υ is optimal. Fortunately, we can use *learning techniques* (read "statistical methods") to reliably estimate this distribution, and then use these estimates to compute a near-optimal approximation; see Subsection 2.3.

Hill-Climbing. Second, even given this distribution P, the task of finding an optimal approximation is intractable; this is the essence of Issue 2 (Intractable Compilation). The ADComp process, defined below, avoids this problem by *hill-climbing* in the space of horn approximations, climbing from some initial approximation to successively better ones, until reaching a local peak.

2.2 ADComp's Behavior

The basic code for the ADComp algorithm appears in Figure 2. Its inputs are an initial theory Σ, error and confidence parameters $\epsilon, \delta > 0$, and a resource bound, the polynomial function $K(\cdot)$. Its output is a near-optimal approximation $\langle S_n, W_m\rangle$, as specified below. ADComp observes a sequence of queries,[6] printing out an answer (`Yes`, `No` or `IDK`) to each, as it computes its approximations of Σ.

ADComp makes use of a particular set of transformations, $\mathcal{T}^S \cup \mathcal{T}^W$, each mapping approximations to approximations. Subsections 2.4 and 2.5 define these transformations more precisely; for now just observe that each $\tau^S \in \mathcal{T}^S$ maps strengthenings to strengthenings and the set $\mathrm{NeighS}[S] = \{\tau^S(S) \mid \tau^S \in \mathcal{T}^S\}$ defines S's neighbors. Similarly, each $\tau^W \in \mathcal{T}^W$ maps weakenings to weakenings, and W's neighbors are $\mathrm{NeighW}(W) = \{\tau^W(W) \mid \tau^W \in \mathcal{T}^W\}$.

In essence, ADComp first computes an initial S_1 and then climbs from S_1 to one of its neighbors, $S_2 \in \mathrm{NeighS}[S_1]$, if S_2 is statistically likely to be superior to S_1, based on the sequence of observed queries. This constitutes one hill-climbing step; in general, ADComp will perform many such steps, climbing from S_1 to S_2 to S_3, etc., until reaching a near optimal S_n. In parallel with this process, ADComp also uses these queries to hill-climb from an initial (computed) W_1 to a neighbor $W_2 \in \mathrm{NeighW}[W_1]$, and then on to $W_3 \in \mathrm{NeighW}[W_2]$, etc., until reaching a near optimal W_m. ADComp returns the resulting $\langle S_n, W_m\rangle$, whose expected score is, with probability at least $1 - \delta$, at an "ϵ-local optimum" with respect to these transformations $\mathcal{T}^S \cup \mathcal{T}^W$:

Theorem 1 *The* $\mathrm{ADComp}(\Sigma, \epsilon, \delta, K(\cdot))$ *process incrementally produces a series of weakenings* $\langle W_1, W_2, \ldots, W_m\rangle$ *and (independently) a sequence of strengthenings* $\langle S_1, S_2, \ldots, S_n\rangle$ *such that, with probability at least* $1 - \delta$,

1. *each successive approximation has an expected score that is strictly better than its predecessor's; i.e.,*
$$\mathrm{C}[\langle S_{i+1}, W_j\rangle] > \mathrm{C}[\langle S_i, W_j\rangle]$$
$$\mathrm{C}[\langle S_i, W_{j+1}\rangle] > \mathrm{C}[\langle S_i, W_j\rangle]$$
and
2. *the final approximation* $\langle S_n, W_m\rangle$ *is an* ϵ*-local optimum; i.e., its expected score is within* ϵ *of the best expected score among its neighbors:*
$$\forall \tau \in \mathcal{T}^S: \;\; \mathrm{C}[\langle S_n, W_m\rangle] \geq \mathrm{C}[\langle \tau(S_n), W_m\rangle] - \epsilon$$
$$\forall \tau \in \mathcal{T}^W: \;\; \mathrm{C}[\langle S_n, W_m\rangle] \geq \mathrm{C}[\langle S_n, \tau(W_m)\rangle] - \epsilon.$$

Moreover, ADComp *requires only polynomial time (and hence only a polynomial number of samples) to decide whether to move from* S_i *to* S_{i+1} *(resp., from* W_j *to* W_{j+1}*) or terminate with a final* S_n *(resp., with a final* W_m*).* □.

(The proof appears in Appendix A.)

2.3 Statistical Foundations

Notice first that $\mathrm{C}[\langle S_i, W_j\rangle] = \mathrm{D}[W_j] + (1 - \mathrm{D}[S_i])$, where

$$\mathrm{D}[T] = E[d(T, \sigma)] = \sum_{\sigma \in \mathcal{Q}} P(\sigma) \times d(T, \sigma)$$

is the likelihood that the theory T will entail a query, over the distribution of queries. As we are only considering transformations that affect only one of W_j or S_i, an approximation $\langle S_i, W_j\rangle$ is, with probability at least

[6] These queries are from the user of the performance system, who is posing queries relevant to one of his application. In general, we need only assume that he is drawing these queries from a stationary distribution, and that this is the same distribution that will be used later, when the resulting performance system $PS^{\alpha}(S_n, W_m, \Sigma)$ is actually being used.

Algorithm ADCOMP(Σ, δ, ϵ, $K(\cdot)$)

Init: $j \leftarrow 0$, $\mathtt{S}_1 \leftarrow$ INITIALS(Σ_H, Σ_N), $\langle \mathtt{W}_1, \Omega_1 \rangle \leftarrow$ INITIALWN$(\Sigma_H, \Sigma_N, K(\Sigma))$,
`FoundGoodS` $\leftarrow$ `False`, `FoundGoodW` $\leftarrow$ `False`

Loop

- $j \leftarrow j+1$, NEIGHS $\leftarrow \{\tau_k^S(\mathtt{S}_j)\}_k$, NEIGHW $\leftarrow \{\tau_k^W[\mathtt{W}_j, \Omega_j](\mathtt{W}_j)\}_k$

$$n_j \leftarrow \left\lceil \frac{2}{\epsilon^2} \ln \frac{2 j^2 \pi^2 \max\{|\text{NEIGHS}|, |\text{NEIGHW}|\}}{3\delta} \right\rceil \tag{1}$$

`/* Get Samples, Print Answers */`

Get n_j samples, $\mathtt{Q}_j = \{\sigma_1, \sigma_2, \ldots, \sigma_{n_j}\}$ from the user

For each sample $\sigma_i \in \mathtt{Q}_j$ do

If for some $\mathtt{W}' \in \{\mathtt{W}_j\} \cup$ NEIGHW, $\mathtt{W}' \models \sigma_i$
then Print "σ_i : Yes"

ElseIf for some $\mathtt{S}' \in \{\mathtt{S}_j\} \cup$ NEIGHS, $\mathtt{S}' \not\models \sigma_i$
then Print "σ_i : No"

Else Print "σ_i : IDK"

End For

`/* Iterate or Terminate, wrt Strengthenings */`

If ¬`FoundGoodS` then

If for some $\mathtt{S}' \in$ NEIGHS,

$$d(\mathtt{S}_j, \mathtt{Q}_j) - d(\mathtt{S}', \mathtt{Q}_j) \geq \frac{\epsilon}{2} \tag{2}$$

then $\mathtt{S}_{j+1} \leftarrow \mathtt{S}'$,

Else `/* Here,` $d(\mathtt{S}_j, \mathtt{Q}_j) - d(\mathtt{S}', \mathtt{Q}_j) < \frac{\epsilon}{2}$ `for all` $\mathtt{S}' \in$ NEIGHS `*/`

`FoundGoodS` $\leftarrow$ `True`

$\mathtt{S}_{final} \leftarrow \mathtt{S}_j$

End If

`/* Iterate or Terminate, wrt Weakenings */`

If ¬`FoundGoodW` then

If for some $\mathtt{W}' \in$ NEIGHW,

$$d(\mathtt{W}', \mathtt{Q}_j) - d(\mathtt{W}_j, \mathtt{Q}_j) \geq \frac{\epsilon}{2} \tag{3}$$

then $\mathtt{W}_{j+1} \leftarrow \mathtt{W}'$, $\Omega_{j+1} \leftarrow$ UPDATEN$(\mathtt{W}_j, \Omega_j)$

Else `/* Here,` $d(\mathtt{W}', \mathtt{Q}_j) - d(\mathtt{W}_j, \mathtt{Q}_j) < \frac{\epsilon}{2}$ `for all` $\mathtt{W}' \in$ NEIGHW `*/`

`FoundGoodW` $\leftarrow$ `True`

$\mathtt{W}_{final} \leftarrow \mathtt{W}_j$

End If

Until: `FoundGoodS` & `FoundGoodW`

Return $PS^{IDK}(\mathtt{S}_{final}, \mathtt{W}_{final}, \Sigma)$

End ADCOMP

Figure 2: Basic ADCOMP algorithm (see description in Subsection 2.2)

$1-\delta$, within ϵ of a local optimum if W_j is within ϵ of a locally optimal weakening, and S_i is within ϵ of a locally optimal strengthening, each with probability at least $1-\frac{\delta}{2}$. We can therefore decouple the task of finding a good strengthening from that of finding a good weakening, and handle each separately.

We would like ADCOMP to climb from a current S_j to a new $S_{j+1} \in \text{NEIGHS}[S_j]$ if S_{j+1} is statistically likely to be strictly better than S_j. (Similarly, from W_i to $W_{i+1} \in \text{NEIGHW}[W_i]$, etc.) The next subsections define appropriate sets of transformations, $\mathcal{T}^S$ and $\mathcal{T}^W$; the rest of this subsection specifies when to make each such transition.

When is S_α better than S_β? By definition, S_α is better than S_β whenever $\text{D}[S_\alpha] < \text{D}[S_\beta]$, or equivalently, when $\text{D}[S_\beta] - \text{D}[S_\alpha] > 0$. The value of $\text{D}[S_\beta] - \text{D}[S_\alpha]$ depends on the distribution, P, which unfortunately is unknown.

We can however use a set of samples to estimate this quantity, and then use statistical methods to bound our confidence of the accuracy of these estimates. To do this, let the variable $\Delta_i = d(S_\beta, \sigma_i) - d(S_\alpha, \sigma_i)$ be the difference in the coverage between S_β and S_α, for the query σ_i. As each query is selected according to a fixed distribution, these Δ_is are independent, identically distributed random variables whose common mean is $\mu = \text{D}[S_\beta] - \text{D}[S_\alpha]$, which is the quantity we want to estimate. Now let

$$\begin{aligned} Y_n &= \frac{1}{n}\sum_{i=1}^{n} d(S_\beta, \sigma_i) - d(S_\alpha, \sigma_i) \\ &= d(S_\beta, \{\sigma_i\}_{i=1}^n) - d(S_\alpha, \{\sigma_i\}_{i=1}^n) \end{aligned}$$

be the sample mean of n samples.[7] This average will tend to the population mean μ as $n \to \infty$; *i.e.*, $\mu = \lim_{n\to\infty} Y_n$. Chernoff bounds [Che52] provide the probable rate of convergence: the probability that "Y_n is more than $\mu + \beta$" goes to 0 exponentially fast as n increases; and, for a fixed n, exponentially as β increases. Formally,[8]

$$\begin{aligned} Pr[Y_n > \mu + \beta] &\leq e^{-2n\beta^2} \\ Pr[Y_n > \mu - \beta] &\leq e^{-2n\beta^2} \end{aligned} \tag{4}$$

The ADCOMP algorithm uses these formulae and the observed values of various $d(S_j, \sigma)$ and $d(\tau_k^S(S_j), \sigma)$, to determine both how confident we should be that $\text{D}[S_j] > \text{D}[S']$ and also whether any "$\mathcal{T}^S$-neighbor" of S_j (*i.e.*, any $\tau_k^S(S_j)$) is more than ϵ better than S_j. (Of course, similar conditions apply for strengthenings: W_α is better than W_β whenever $\text{D}[W_\alpha] - \text{D}[W_\beta] > 0$, etc.) See the proof in Appendix A.

2.4 Finding a good Horn Strengthening

A "horn-strengthening" of the clause $\gamma = \{a_1, \ldots, a_k, \neg b_1, \ldots, \neg b_\ell\}$ is any maximal clause that is a subset of γ and is Horn — *i.e.*, each horn-strengthening is formed by simply discarding all but one of the positive literals. Here, there are k horn strengthenings of this γ, each of the form $\gamma_j = \{a_j, \neg b_1, \ldots, \neg b_\ell\}$. (*E.g.*, the 2 horn strengthening of the non-Horn clause $\gamma \equiv a \vee b \vee \neg c \vee \neg d$ are $\gamma_1 \equiv a \vee \neg c \vee \neg d$ and $\gamma_2 \equiv b \vee \neg c \vee \neg d$.)

We can write $\Sigma = \Sigma_H \cup \Sigma_N$, where each element of Σ_H is a Horn clause, and each element of $\Sigma_N = \{\gamma^i\}_{i=1}^m$ is a non-Horn clause. [SK91] proves that each weakest strengthening is of the form $S_o = \Sigma_H \cup \Sigma'_N$, where $\Sigma'_N = \{\gamma^{i'}\}_{i=1}^m$ such that each $\gamma^{i'} \in \Sigma'_N$ is a horn-strengthening of some $\gamma^i \in \Sigma_N$. By identifying each Horn-strengthened theory with the "index" of the positive literal used (*i.e.*, $\gamma^i{}_j = \{a_j^i, \neg b_1^i, \ldots, \neg b_{\ell(i)}^i\}$), we can consider any Horn-strengthened theory to be a set of the form $S_{\langle j(1), j(2), \ldots, j(m)\rangle} = \Sigma_H \cup \{\gamma^1_{j(1)}, \gamma^2_{j(2)}, \ldots, \gamma^m_{j(m)}\}$. Notice that each of these strengthenings S_i is "small", in fact, $|S_i| = |\Sigma|$.

We can navigate about this space of Horn-strengthened theories by incrementing or decrementing the index of a specific non-Horn clause: That is, define the set of $2m$ transformations $\mathcal{T}^S = \{\tau_k^+, \tau_k^-\}_{k=1}^m$ where each τ_k^+ (resp., τ_k^-) is a function that maps one strengthening to another, by incrementing (resp., decrementing) the "index" of k^{th} clause — *e.g.*, $\tau_k^+(S_{\langle 3,9,\ldots,i_k,\ldots,5\rangle}) = S_{\langle 3,9,\ldots,i_k+1,\ldots,5\rangle}$, and $\tau_i^-(S_{\langle 3,9,\ldots,i_k,\ldots,5\rangle}) = S_{\langle 3,9,\ldots,i_k-1,\ldots,5\rangle}$. (Of course, the addition and subtraction operations wrap around.)

The ADCOMP process, therefore, starts with an arbitrary horn-strengthening — here, the $S_{\langle 1,1,\ldots,1\rangle}$ returned by $\text{INITIALS}(\Sigma_H, \Sigma_N)$ — and then hill-climbs in this space of horn-strengthened theories, using the set of $\mathcal{T}^S$ transformations defined above. It will terminate on reaching an S_j which is an ϵ-local optimum. (Notice this S_j is not necessarily a weakest strengthening.)

2.5 Finding a good Horn Weakening

[SK91] proves that there is a unique optimal weakening, w_s, and presents the LUB algorithm for computing it. Their algorithm is equivalent to $\text{INITIALWN}(\Sigma_H, \Sigma_N, \infty)$, using the process shown in Figure 3. The final $w_s = \text{INITIALWN}(\Sigma_H, \Sigma_N, \infty)$ is the set of all horn implicates of the initial theory. It is easy to see that this w_s will have the largest possible $\text{D}[\cdot]$ value over all weakenings, for any distribution. Unfortunately, it can also be exponentially larger than the original theory [KS92].

As mentioned above, we avoid this potential blow-up by considering only weakenings of size at most $K = K(\Sigma)$, where $K(\cdot)$ is a user-supplied polynomial function.[9] Our goal, therefore, is to find the weakening of this size that is maximally categorical, over the distribution of queries.

ADCOMP performs a (tractable) hill-climbing search through the space of K-sized Horn weakenings of Σ, attempting to find one that has good empirical coverage (an ϵ-locally optimal expected score). As we are

[7] Notice $d(T, Q) = \frac{1}{|Q|}\sum_{\sigma\in Q} d(T, \sigma)$, for any theory T and any set of queries Q.

[8] See [Bol85, p. 12]. *N.b.*, these inequalities holds for essentially *arbitrary distributions*, not just normal distributions, subject only to the minor constraint that the random variables $\{d_i\}$ be bounded.

[9] To avoid degeneracies, we will assume that $K(\Sigma) \geq |\Sigma|$.

```
Algorithm INITIALWN( Σ_H, Σ_N, K)
  W ← Σ_H;  Ω ← Σ_N;     Done ← True;     j ← 0
  Repeat
    j ← j + 1
    For each w ∈ W, and each n ∈ Ω
      If w and n resolve
      Then Let λ be the resolvent of w and n
           If    /* λ is NOT subsumed by any clause in W ∪ Ω */
                 ∀α ∈ W ∪ Ω :  α ⊈ λ
           Then Done ← False
                    /* Remove from W, Ω all clauses that λ subsumes */
                W ← {w ∈ W | λ ⊈ w}
                Ω ← {n ∈ Ω | λ ⊈ n}
                If λ is horn,
                  Then /* add λ to W */
                       W ← W ∪ {λ}
                  Else /* λ is non horn; add λ to Ω */
                       Ω ← Ω ∪ {λ}
                End If
           End If
      End If
    End For
  Until  Done  or  |W| = K  or  |Ω| = K  or  j = K
  Return ⟨W, Ω⟩
End INITIALWN
```

Figure 3: INITIALWN Algorithm, adapted from LUB in [SK91, p907]

only considering reasonably-sized theories, the result of this search is a useful Horn weakening of Σ from which we can perform *tractable* inference, thus addressing ISSUE 4 (EXPONENTIALLY LARGE WEAKENING).

ADCOMP uses the INITIALWN algorithm to generate an initial bounded weakening $\langle W_1, \Omega_1 \rangle$ = INITIALWN(Σ_H, Σ_N, K). (Notice this process is efficient, as INITIALWN will perform at most K iterations.) ADCOMP then uses a "1-step variant" of INITIALWN to climb to successive weakenings. In particular, given $\langle W_j, \Omega_j \rangle$ at iteration j, ADCOMP will consider climbing from W_j using the transformations[10] $\mathcal{T}^W[W_j, \Omega_j] = \{ \tau_{h_1,n_1,h_2} \}_{h_1,h_2 \in W_j; n_1 \in \Omega_j}$, where $\tau_{h_1,n_1,h_2}(W_j)$ returns

- $\{\}$, if h_1 does not resolve with n_1.
  Otherwise, let λ be the result of resolving h_1 with n_1.
- $\{\}$, if λ is not horn, or if λ is subsumed by any element of W_j.
  Otherwise,
- $W_j \cup \{\lambda\} - \{\eta_k\}_k$, if λ is horn and subsumes each $\eta_k \in W_j$. (Of course, there must be at least one such η.)
  Otherwise,
- $W_j \cup \{\lambda\}$, if $|W_j| < K$.
  Otherwise,
- $W_j \cup \{\lambda\} - \{h_2\}$, if λ is horn and does not subsume any clause in W_j (*i.e.*, τ_{h_1,n_1,h_2} replaces h_2 with λ in W_j). □.

[10] We write the set of transformations as $\mathcal{T}^W[W_j, \Omega_j]$ to indicate that it depends on the current weakening and its non-horn complement, $\langle W_j, \Omega_j \rangle$ and so can change from one weakening W_j to the next, W_{j+1}.

The resulting set of weakenings

$$\text{NEIGHW}(W_j) = \left\{ W \;\middle|\; \begin{array}{l} W = \tau_{h_1,n_1,h_2}(W_j) \;\&\; \\ \tau_{h_1,n_1,h_2} \in \mathcal{T}^W[W_j, \Omega_j] \\ \&\; W \neq \{\} \end{array} \right\}$$

includes all and only the non-$\{\}$ values $\tau_{h_1,n_1,h_2}(W_j)$.

Example: Imagine INITIALWN returned the initial pair

$$\begin{array}{rcl} W_1 & = & \{ \neg a,\ \neg b,\ d \} \\ \Omega_1 & = & \{ a \vee b \vee \neg c,\ a \vee c \vee \neg d \} \end{array}$$

and let $K = 3$, meaning W_1 is filled to its capacity. Here, there are $|W_1| \times |\Omega_1| \times |W_1| = 3 \times 2 \times 3 = 18$ different transformations:

$$\mathcal{T}^W[W_1, \Omega_1] = \left\{ \begin{array}{lll} \tau_{\neg a,\, a\vee b\vee\neg c,\, \neg a} & \tau_{\neg a,\, a\vee b\vee\neg c,\, \neg b} & \tau_{\neg a,\, a\vee b\vee\neg c,\, d} \\ \tau_{\neg a,\, a\vee c\vee\neg d,\, \neg a} & \tau_{\neg a,\, a\vee c\vee\neg d,\, \neg b} & \tau_{\neg a,\, a\vee b\vee\neg d,\, d} \\ \tau_{\neg b,\, a\vee c\vee\neg d,\, \neg a} & \cdots & \\ & \cdots & \tau_{d,\, a\vee c\vee\neg d,\, d} \end{array} \right\}$$

Notice most transformations are degenerate, simply returning $\{\}$ — including all 3 of the form $\tau_{d,\, a\vee b\vee\neg c,\, \cdot}(W_1) = \{\}$ as d does not resolve with $a \vee b \vee \neg c$. The three transformations $\tau_{d,\, a\vee c\vee\neg d,\, \cdot}$ are also degenerate, as the resolvent of d and $a \vee c \vee \neg d$, namely $a \vee c$, is not horn. (But see Issue 5 in Subsection 3.)

To illustrate a non-degenerate transformation, observe $\tau_{\neg a,\, a\vee b\vee\neg c,\, d}(W_1) = \{b \vee \neg c,\, \neg a,\, \neg b\}$. Here, as $|W_1| = K$, we had to remove one element of W_1, namely d, to make space for $b\vee\neg c$, the resolvent of $\neg a$ and $a\vee b\vee\neg c$. If K had been larger, then $\tau_{\neg a,\, a\vee b\vee\neg c,\, d}(W_1)$ could simply add in this new $b \vee \neg c$, producing the 4-*element* weakening $\{b \vee \neg c,\, \neg a,\, \neg b,\, d\}$. □

If one of the $W' = \tau_{h_1,n_1,h_2}(W_j) \in \text{NEIGHW}(W_j)$ neighbors passes Equation 3 and becomes the new "current weakening" W_{j+1}, ADCOMP will use the UPDATEN process to compute a new Ω_{j+1}. UPDATEN first resolves each clause in the new W_{j+1} with each clause in Ω_j, then forms Ω_{j+1} by adding all the unsubsumed *non-Horn* clauses to Ω_j, and removing all subsumed clauses. (This is like the set of $\mathcal{T}^W[W_j, \Omega_j] = \{\tau_{h_1,n_1,h_2}\}$ transformations, but adding non-horn clauses to Ω_j, rather than horn clauses to W_j.) To keep $|\Omega_{j+1}| \leq K$, UPDATEN may have to delete an existing clause from Ω_j before adding a new resolvent. (This choice is arbitrary; we could, for example, simply remove the "oldest" clauses in Ω_j until the size bound is reached. Of course, there are many other approaches.)

Example: To continue with the earlier example, suppose $\tau_{\neg a,\, a\vee b\vee\neg c,\, d}(W_1) = \{b \vee \neg c,\, \neg a,\, \neg b\}$ passes Equation 3 and so becomes W_2. To compute Ω_2, UPDATEN first resolves each $h \in W_2$ with each $n \in \Omega_1$ (producing $\{\ b \vee \neg c,\ a \vee \neg c,\ a \vee b \vee \neg d,\ c \vee \neg d\ \}$), then adds to Ω_1 the non-horn propositions, forming $\{\ a \vee b \vee \neg c,\ a \vee c \vee \neg d,\ a \vee b \vee \neg d\ \}$. It then removes all subsumed clauses, leaving $\Omega_2 = \{\ a \vee c \vee \neg d,\ a \vee b \vee \neg d\ \}$. □

Notice each resulting theory W' can have no more clauses than W_j; hence, $|W'| \leq |W_j| \leq K$. Moreover, there are only $O(K^3)$ possible transformations and each of these new weakenings can be computed efficiently.

2.6 Efficiency

Each of ADCOMP's individual steps is tractable: The only potentially problematic steps involve asking whether $T \models^? \sigma$, where T is S_j, $\tau^S(S_j)$, W_j or $\tau^W(W_j)$. However, as each of these theories is horn and of bounded size (at most K), each of these computations is efficient.

As an important aside, notice that ADCOMP is using this battery of efficient tests to approximate the intractable $\Sigma \models^? \sigma$ test: correctly concluding that $\Sigma \models \sigma$ whenever either $W_j \models \sigma$ or $\tau^W(W_j) \models \sigma$ for any $\tau^W \in \mathcal{T}^W$, and that $\Sigma \not\models \sigma$ whenever $S_i \not\models \sigma$ or $\tau^S(S_i) \not\models \sigma$ for any $\tau^S \in \mathcal{T}^S$. ADCOMP will return IDK only if *none* of these tests succeeds — *i.e.*, if $W_j \not\models \sigma$ and $\tau^W(W_j) \not\models \sigma$ for all $\tau^W \in \mathcal{T}^W$, $S_i \models \sigma$ and $\tau^S(S_i) \models \sigma$ for all $\tau^S \in \mathcal{T}^S$.

ADCOMP can perform at most $|\mathcal{T}^W[W_j, \Omega_j]| + |\mathcal{T}^S|$ such derivations for each sample query, which is also a polynomial in the relevant parameters. Observe, moreover, that each iteration of the ADCOMP process can involve only a polynomial number of samples (n_j from Equation 1). The only part of this process that is not necessarily bounded by a polynomial is the number of iterations required. However, this is not necessarily problematic, as ADCOMP is essentially an anytime system [DB88], returning successively better and better horn approximations.

In fact, our ADCOMP can be viewed as a natural extension of the anytime compiler discussed in [SK91], as each system runs in parallel with a performance system that is using the current best approximation to return responses to the queries presented. ADCOMP differs (1) by using the set of observed queries to *guarantee* with provably high probability that each of the approximations truly is an improvement over its predecessors; (2) by avoiding the intractable $\Sigma \models \sigma$ test while learning; and (3) by guaranteeing that the approximations produced will admit tractable inference.

3 Extensions

This section discusses various extensions to our basic approach and the ADCOMP algorithm shown in Figure 2.

Extension 1. Minor Adjustments: ADCOMP uses a single value of K to bound the sizes of W_j and Ω_j and as the time limit for the INITIALWN process. An obvious variant would permit the user to supply several values, to separately specify the different size constraints and time bound.

Notice also that ADCOMP could perform a quick post-processing on the final strengthening S_n (resp., final weakening W_m), to convert this horn theory into a possibly smaller horn theory by resolving its clauses together and removing all subsumed expressions.

Extension 2. ADCOMP works in "batch" mode — using a collection of n_j (Equation 1) samples to decide whether to iterate from S_j to S_{j+1} or to stop improving the strengthening, and also whether to iterate from W_j to W_{j+1}, etc. [GJ92] presents PALO, a related algorithm (but designed for a different task) that can make these decisions after each individual sample. PALO can potentially require fewer samples on each iteration than ADCOMP, as PALO will consider climbing to a (probabilistically) better element or terminating, after seeing each sample. We have designed an algorithm, ADCOMP*, that basically uses PALO's techniques, but handles ADCOMP's application, and confirmed that ADCOMP* does satisfy Theorem 1.[11]

Extension 3. In general, ADCOMP must compute the values of $d(W', \sigma) - d(W_j, \sigma)$ for each W' that is a $\mathcal{T}^W$-neighbor of W_j. We can always obtain this information by constructing these neighboring W's, and using them to compute the relevant values of $d(W', \sigma)$. Alternatively, there are often ways of computing these values, based only on running the original W_j. As an example, imagine that $W_j \models \sigma$, and let $\{h_i\} \subseteq W_j$ be σ's support in W_j (*i.e.*, $\{h_i\} \models \sigma$). Clearly $W' \models \sigma$ will hold for each $W' \in \text{NEIGHW}(W_j)$ whenever $\{h_i\} \subseteq W'$ as well. Hence, we can guarantee that $d(W_j, \sigma) - d(W', \sigma) = 0$

[11] We chose to present the simpler ADCOMP version for pedagogic reasons, as ADCOMP* is much more difficult to explain.

in this context. (Of course, this same idea also applies to computing the values of $d(S_i, \sigma) - d(S', \sigma)$.)

Extension 4. We can consider using other transformations, especially when seeking an optimal weakening. For example, [KS92] suggests a way of shrinking the size of some horn weakening by adding new vocabulary terms to the initial theory; it would be easy to also include transformations that implement this idea. Other recent papers, including [DE92], propose other techniques for finding good (not necessarily horn) approximations.

Extension 5. The set of $\mathcal{T}^W$ transformations described in Subsection 2.5 will not always allow ADCOMP to explore the entire space of K-sized weakenings. Consider, for example, the theory $\Sigma = W_1 \cup \Omega_1$, where

$$\begin{aligned} W_1 &= \{\neg a \vee c,\ \neg b \vee c\} \\ \Omega_1 &= \{a \vee b\}\ . \end{aligned}$$

Notice that all transformations in $\mathcal{T}^W[W_1, \Omega_1]$ are degenerate, as there are only two resolvents of elements in W_1 with elements in Ω_1 (*viz.*, $a \vee b$ and $b \vee c$) and neither is horn. As W_1 has no neighbors, ADCOMP cannot consider any alternative weakenings, meaning it will necessarily miss the superior weakening, $W_{opt} = \{c\}$.

However, notice that we could have reached this W_{opt} weakening in two steps, if we had used transformations that could produce new non-horn clauses. Given such transformations, we could then form the new pair $\langle W_2, \Omega_2 \rangle$, where $W_2 = W_1$, and $\Omega_2 = \{\, a \vee b,\ b \vee c,\ a \vee c \,\}$. Now, by resolving the clauses in W_2 with those in Ω_2, we would produce the desired $W_3 = \{c\}$ (along with $\Omega_3 = \{\, b \vee c \,\}$).

Unfortunately, there is a basic problem with this approach: The score of any weakening/non-horn-complement pair $\langle W, \Omega \rangle$ depends only on the observed categoricity of the weakening part W (*i.e.*, on the values of $d(W, \sigma)$ used to approximate $\mathrm{D}[W]$). This means that the score of $\langle W, \Omega' \rangle$ is necessarily the same as the score of $\langle W, \Omega \rangle$, even though Ω' is different from Ω. Thus, $\langle W, \Omega' \rangle$ can never be strictly better than $\langle W, \Omega \rangle$, and so ADCOMP will never climb to it. This is why ADCOMP does not even generate these equal-cost neighbors.

There is an obvious alternative. Given $\langle W, \Omega \rangle$, the alternative ADCOMP$_1$ algorithm produces new non-horn components, $\{\Omega_i\}$, as well as new weakenings, $\{W_j\}$. Just like ADCOMP, this algorithm also compares the values of $d(W, Q)$ with each $d(W_j, Q)$, over a prescribed set of queries Q. If any $W' \in \{W_j\}$ passes the Equation 3 test, ADCOMP$_1$ will climb to this new weakening. Otherwise, if none of the alternative weakenings $\{W_j\}$ looks much better, ADCOMP$_1$ will randomly pick one of the alternative non-horn theories, $\Omega' \in \{\Omega_i\}$, and climb "sideways" to the weakening-pair $\langle W, \Omega' \rangle$. This produces a new neighborhood — a different set of neighboring weakenings $\{W'_j\}$ and of neighboring non-horn components $\{\Omega'_i\}$. ADCOMP$_1$ will then compare Ws score with each its neighbors, and climb to an $W'' \in \{W'_j\}$ if $d(W'', Q')$ is sufficiently better than $d(W, Q')$ for the (new) set of sample queries Q'. If none qualify, ADCOMP$_1$ will again walk side-ways, to one of the neighboring $\Omega'' \in \{\Omega'_i\}$; and so forth.

Of course, we may not want to wander about on this equal-score plateau forever. The ADCOMP$_2$ variant will permit only `MaxPlateauWalks` steps before terminating its search for a good weakening, where `MaxPlateauWalks` $\in \mathcal{Z}^+$ is a user-specified parameter. Another variant is ADCOMP$_3$: If none of the W_js appears better, ADCOMP$_3$ will stochastically decide whether to walk to a new $\Omega' \in \{\Omega_i\}$ (with probability `PlateauWalkProb`) or to terminate. Here, this `PlateauWalkProb` $\in [0, 1]$ is a user-specified parameter.

Each of these three variants will have to prevent looping (*i.e.*, walking from Ω_1 to Ω_2 to ...and back to Ω_1), perhaps by imposing some ordering on the Ω_i theories, and only going from Ω_i to a new Ω_{i+1} with a strictly larger value. Also, each variant may use some bias on the set of Ω_is, to prefer some over others.

Extension 6. The ADCOMP process uses the function $K(\cdot)$ to bound the size of the weakening. In essence, this function quantifies how much time the user will allow the system to spend in answering a query, before insisting that it stop and return IDK. Hence, by selecting an appropriate $K(\cdot)$ function, the user can direct ADCOMP to the class of approximations that optimizes his implicit utility measure which embodies a particular tradeoff between efficiency and categoricity.

In general, the user may want to use a more general measure for ranking different approximations, which can depend on other factors as well. For example, is complete accuracy important? If not, how does it tradeoff with time concerns? Is incompleteness (in the form of using "IDK") better than errors? ...or are these two equally bad?

We can provide the user with greater flexibility by allowing him to specify his own scoring function, $c_i : \mathrm{Approx}_\infty(\Sigma) \times \mathcal{Q} \mapsto \Re$, where $c_i(\Upsilon, \sigma)$ indicates how well the approximation Υ does at solving σ.[12] Following [GE91], we allow this scoring function to be a combination of various factors, including accuracy, categoricity and efficiency. Given any such c_i, our goal is to find the performance system (call it PS_{c_i}) whose expected c_i score is maximal.

To illustrate the range of possible scoring functions, we can, for example, define c_0 to be a scoring function that imposes a very high penalty for incorrect or incomplete answers. If this penalty is sufficiently high, the optimal PS_{c_0} system will necessarily be sound and complete with respect to Σ, meaning it will have to use $\alpha = SND$. The ADCOMP system presented above implicitly uses a different function, call it c_1, that does not impose as severe a restriction: As it does not insist on completeness but does require (poly-time) efficiency, c_1 prefers performance systems that return IDK if the approximation does not cover the query. Hence, the c_1-optimal performance system uses $\alpha = IDK$ rather than $\alpha = SND$, in addition to imposing a limitation on the size of the approximations.

A slightly different criterion, encoded by the c_2 function, would explicitly use both computational time and

[12] The set $\mathrm{Approx}_\infty(\Sigma) = \bigcup_{k=1}^{\infty} \mathrm{Approx}_k(\Sigma)$ includes *all* horn approximations, of arbitrary size.

categoricity to rank approximations: $c_2(\langle S, W\rangle \sigma)$ is the time $PS = PS(S, W, \Sigma)$ spends answering σ if PS finds a definite answer, or a large negative value, B, if PS returns IDK. Here the size of B could indirectly determine, for example, whether $\alpha = IDK$ was better than $\alpha = SND$, and also the allowed size of the W set.

This suggests a way of addressing ISSUE 1 (WHICH α?): *viz.*, by using a variant, call it ADCOMP$_\alpha$, that searches for a good α-function in parallel with its search for good weakenings and strengthenings. The particular c_i function specified will determine whether the optimal performance system should use $\alpha = SND$ or $\alpha = GUE$ or $\alpha = IDK$, or possibly some other pre-defined α function. (See [GS92].)

Future Work. First, we are currently implementing ADCOMP and plan to test it empirically in order to determine just how categorical and efficient it really is, both on real world problems and on "hard" cases [MSL92]. We also plan to compare ADCOMP with other theorem proving processes, including incomplete systems like GSAT [SLM92]. We anticipate that these studies will give us insights on many of the issues mentioned above, including the different ways of handling the "plateau problem" mentioned in Extension 5. Second, we are currently looking for a less sample-hungry variant of ADCOMP, perhaps based on a statistical stopping criterion that is less generous than the Chernoff bounds (Equation 4). A third extension is to find better sets of transformations, especially for moving about the space of horn weakenings. We will also consider other types of non-*Horn* approximations: syntactically defined formula from which derivation is efficient. The final task is to extend these ideas from propositional logic to full first-order predicate calculus.

4 Conclusion

It is often critical for an agent to use its knowledge to produce answers efficiently. Unfortunately, this task is intractable in general. [SK91] presents a way around this problem, describing an algorithm that produces a semantically similar Horn approximation from which many queries can be answered efficiently. Our paper extends that work by providing a more utilitarian objective function, one that prefers approximations that produce answers efficiently to most *anticipated* queries. It then defines the ADCOMP algorithm, that effectively finds near optimal approximations: ADCOMP uses a set of queries to estimate the distribution of anticipated queries, then uses this information to hill-climb in the space of possible approximations until reaching one that is, with high probability, near a local optimum. We also discuss various extensions to this algorithm, to permit it to use a more general user-specified utility measure, which can embody the user's tradeoffs between efficiency and accuracy, etc.

Our approach differs from earlier approaches to theory approximation in a number of significant ways: [1] ADCOMP uses a set of observed queries to "learn" an approximation that is essentially at a local optimum within the class of possible approximations; [2] this near-optimal approximation is guaranteed to perform tractable inference; and [3] ADCOMP learns this near-optimal approximation without computing answers to queries using original theory, meaning it can find effective approximations efficiently.

A Proof of Theorem 1

Theorem 1 *The* ADCOMP$(\Sigma, \epsilon, \delta, K)$ *process incrementally produces a series of weakenings* $\langle W_0, W_1, \ldots, W_m\rangle$ *and (independently) a sequence of strengthenings* $\langle S_0, S_1, \ldots, S_n\rangle$ *such that, with probability at least* $1-\delta$,

1. *each successive approximation has an expected score that is strictly better than its predecessor's; i.e.,*
$$C[\langle S_{i+1}, W_j\rangle] > C[\langle S_i, W_j\rangle]$$
$$C[\langle S_i, W_{j+1}\rangle] > C[\langle S_i, W_j\rangle]$$
and
2. *the final approximation* $\langle S_n, W_m\rangle$ *is an* ϵ*-local optimum; i.e., its expected score is within* ϵ *of the best expected score among its neighbors:*
$$\forall\, \tau \in \mathcal{T}^S:\ C[\langle S_n, W_m\rangle] \geq C[\langle \tau(S_n), W_m\rangle] - \epsilon$$
$$\forall\, \tau \in \mathcal{T}^W:\ C[\langle S_n, W_m\rangle] \geq C[\langle S_n, \tau(W_m)\rangle] - \epsilon.$$

Moreover, ADCOMP *requires only polynomial time (and hence only a polynomial number of samples) to decide whether to move from* S_i *to* S_{i+1} *(resp., from* W_j *to* W_{j+1}*) or terminate with a final* S_n *(resp., with a final* W_m*).* □.

Proof: Subsection 2.6 above already established ADCOMP's computational efficiency.

To prove parts 1 and 2 of the theorem: Consider first a single iteration of the ADCOMP algorithm, and consider only the strengthenings. Notice there are two ways that ADCOMP can make a mistake:

1. If some $S' \in$ NEIGHS appears to be better than S_j but is not; or
2. If some $S' \in$ NEIGHS is really more than ϵ better than S_j, but appears not to be.

Let

$$p_1^j = Pr\left[\begin{array}{c}\exists S' \in \text{NEIGHS}.\ d(S_j, Q_j) - d(S', Q_j) \geq \frac{\epsilon}{2} \\ \text{and } \mathrm{D}[S_j] < \mathrm{D}[S']\end{array}\right]$$

$$p_2^j = Pr\left[\begin{array}{c}\exists S' \in \text{NEIGHS}.\ d(S_j, Q_j) - d(S', Q_j) < \frac{\epsilon}{2} \\ \text{and } \mathrm{D}[S'] < \mathrm{D}[S_j] - \epsilon\end{array}\right]$$

be the respective probabilities of these events. Now observe that

$$\begin{aligned} p_1^j &\leq \sum_{S' \in NeighS} Pr\left[\begin{array}{c} d(S_j, Q_j) - d(S', Q_j) \geq \frac{\epsilon}{2} \\ \text{and } \mathrm{D}[S'] - \mathrm{D}[S_j] < 0\end{array}\right] \\ &\leq \sum_{S' \in NeighS} e^{-2n_j\left(\frac{\epsilon}{2}\right)^2} \qquad (5) \\ &\leq |\text{NEIGHS}|\, e^{-2\left(\frac{2}{\epsilon^2}\ln\frac{2j^2\pi^2\max\{|NeighS|,\,|NeighW|\}}{3\delta}\right)\left(\frac{\epsilon}{2}\right)^2} \\ &= |\text{NEIGHS}|\,\frac{3\delta}{2j^2\pi^2\max\{|NeighS|,\,|NeighW|\}} \\ &\leq \frac{1}{j^2}\frac{3\delta}{2\pi^2} \end{aligned}$$

Line 5 uses Chernoff bounds (Equation 4).[13] Similarly,

$$p_2^j \leq \sum_{S' \in NeighS} Pr\left[\begin{array}{c} d(S_j, Q_j) - d(S', Q_j) < \frac{\epsilon}{2} \\ \text{and } \mathrm{D}[S_j] - \mathrm{D}[S'] > \epsilon \end{array} \right]$$
$$\leq \sum_{S' \in NeighS} e^{-2n_j\left(\frac{\epsilon}{2}\right)^2} \leq \frac{1}{j^2}\frac{3\delta}{2\pi^2}$$

Hence, the probability of ever making either mistake at any iteration is under

$$\sum_{j=1}^{\infty} p_1^j + p_2^j \leq \sum_{j=1}^{\infty} \frac{1}{j^2}\frac{3\delta}{2\pi^2} + \frac{1}{j^2}\frac{3\delta}{2\pi^2}$$
$$= \delta\frac{3}{\pi^2}\sum_{j=1}^{\infty}\frac{1}{j^2} = \delta\frac{3}{\pi^2}\frac{\pi^2}{6} = \frac{\delta}{2}$$

The same arguments, *mutatis mutandis*, hold for finding good weakening: the probability of either climbing to an inferior weakening, or stopping at a weakening that is not an ϵ-local optimum, is also bounded by $\delta/2$. Hence, the probability of either making a mistake for the strengthenings, or weakenings, is under $\frac{\delta}{2} + \frac{\delta}{2} = \delta$, as desired. □.

References

[Bol85] B. Bollobás. *Random Graphs.* Academic Press, 1985.

[Che52] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC71*, pages 151–58, 1971.

[DB88] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proceedings of AAAI-88*, pages 49–54, August 1988.

[DE92] Mukesh Dalal and David Etherington. Tractable approximate deduction using limited vocabulary. In *Proceedings of CSCSI-92*, Vancouver, May 1992.

[DG84] William F. Dowling and Jean H. Gallier. Linear time algorithms for testing the satisfiability of propositional horn formula. *Journal of Logic Programming*, 3:267–84, 1984.

[EBBK89] David W. Etherington, Alex Borgida, Ronald J. Brachman, and Henry Kautz. Vivid knowledge and tractable reasoning: Preliminary report. In *Proceedings of IJCAI-89*, pages 1146–52, 1989.

[GE91] Russell Greiner and Charles Elkan. Measuring and improving the effectiveness of representations. In *Proceedings of IJCAI-91*, pages 518–24, Sydney, Australia, August 1991.

[13] This relies on the fact that the distribution of queries is stationary, meaning that, for any given pair of strengthenings S_j and S', the values of the random variables $\Delta_i = d(S_j, \sigma_i) - d(S', \sigma_i)$ are drawn from a stationary distribution.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[GJ92] Russell Greiner and Igor Jurišica. A statistical approach to solving the EBL utility problem. In *Proceedings of AAAI-92*, San Jose, 1992.

[GS92] Russell Greiner and Dale Schuurmans. Learning useful horn approximations. Technical report, Siemens Corporate Research, 1992.

[KS92] Henry Kautz and Bart Selman. Speeding inference by acquiring new concepts. In *Proceedings of AAAI-92*, San Jose, July 1992.

[MSL92] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distribution of sat problems. In *Proceedings of AAAI-92*, San Jose, July 1992.

[SK91] Bart Selman and Henry Kautz. Knowledge compilation using horn approximations. In *Proceedings of AAAI-91*, pages 904–09, Anaheim, August 1991.

[SLM92] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of AAAI-92*, pages 440–46, San Jose, July 1992.

Tractable Deduction in Knowledge Representation Systems

Mukesh Dalal*
Rutgers University
Computer Science Department
New Brunswick, NJ 08903, U.S.A.

Abstract

A widely-used way to deal with the intractability of various deduction problems is based on identifying their tractable cases. Current techniques for this are highly problem specific. We present a new technique that obtains powerful tractability results for many different problems. Our technique is based on a notion of partial consistency for logical theories. Any set of theories for which a fixed level of partial consistency can guarantee logical consistency provides a tractable class of deduction problems. We apply this technique to obtain tractability results in the areas of constraint satisfaction problems, databases with incomplete information and disjunctive logic programming.

1 INTRODUCTION

It is well known that deductive reasoning in a knowledge representation system becomes more difficult as the representation language becomes more expressive [Levesque and Brachman, 1985]. Deductive reasoning (or deduction) is intractable[1] even for the very weak representation language of propositional logic [Cook, 1971]. Since practical knowledge representation systems need more expressive representation languages [Doyle and Patil, 1991], this intractability of deduction is a serious problem.

A widely-used approach to deal with this intractability is based on identifying those representation languages for which deduction is provably tractable. For example, the frame description language $\mathcal{FL}^-$ that allows only AND, OR, and SOME operators has a polynomial time algorithm for detecting subsumption between descriptions [Brachman and Levesque, 1984]. Another well known example is the relational data model that allows only positive atomic facts to be explicitly represented, such that querying such databases is tractable [Codd, 1970]. Although many such tractable cases have been independently identified for various kinds of deduction problems, their descriptions are highly problem-specific and do not seem to apply to one another.

In this paper we present a very general characterization that can be used for identifying tractable cases of many different kinds of deduction problems. We also present some new tractable cases that have been obtained using our characterization. We also present three widely-differing applications to illustrate the power and generality of our approach.

The basic idea behind our approach is as follows. The problem of deduction in knowledge representation is a variant of the consistency problem, i.e., determining whether any given logical theory is consistent. Consistency can be determined easily in absence of any disjunctive information. One way to deal with disjunctions is to use some form of case-analysis, i.e., assuming that some particular proposition is true and then propagating its consequences. It is possible to determine whether a theory is consistent by allowing unlimited nesting of assumptions. By restricting the depth of such nesting, we obtain a notion of partial consistency for logical theories. Any collection of theories for which a fixed level of partial consistency can guarantee logical consistency provides a tractable class of deduction problems. Our notion of partial consistency for logical theories is similar to but much more general than that for constraint satisfaction problems [Freuder, 1978].

The plan of rest of the paper is as follows. In the next section we give some basic definitions and terminology. In Section 3, we present tractable, but incomplete, methods to detect inconsistency and to propagate facts in logical theories. These methods are used

*Email: dalal@cs.rutgers.edu

[1]For the purpose of this paper, a problem is "intractable" if the corresponding decision problem is either NP-Hard or CoNP-Hard [Garey and Johnson, 1979], otherwise the problem is "tractable", "easy" or "simple".

1. $\mathbf{f} \wedge \psi \Rightarrow \mathbf{f}$
2. $\mathbf{f} \vee \psi \Rightarrow \psi$
3. $\mathbf{t} \wedge \psi \Rightarrow \psi$
4. $\mathbf{t} \vee \psi \Rightarrow \mathbf{t}$
5. $\alpha \wedge \psi \Rightarrow \alpha \wedge [\psi]^{\mathbf{t}}_{\alpha}$
6. $\alpha \vee \psi \Rightarrow \alpha \vee [\psi]^{\mathbf{f}}_{\alpha}$
7. $x \neq x \Rightarrow \mathbf{f}$
8. $x = x \Rightarrow \mathbf{t}$
9. $\psi \wedge ((\alpha \wedge \psi_1) \vee (\alpha \wedge \psi_2) \vee \ldots) \Rightarrow \alpha \wedge \psi \wedge (\psi_1 \vee \psi_2 \vee \ldots)$
10. $\psi \vee ((\alpha \vee \psi_1) \wedge (\alpha \vee \psi_2) \wedge \ldots) \Rightarrow \alpha \vee \psi \vee (\psi_1 \wedge \psi_2 \wedge \ldots)$
11. $p(\overline{v}, x, \overline{w}) \wedge \neg p(\overline{v}, y, \overline{w}) \Rightarrow p(\overline{v}, x, \overline{w}) \wedge \neg p(\overline{v}, y, \overline{w}) \wedge x \neq y$
12. $x = y \wedge \psi \Rightarrow x = y \wedge [\psi]^{x}_{y} \quad (x \prec y)$

Figure 1: Rewrite rules for FPF

in Section 4 to obtain levels of partial consistency. We also present an algorithm for achieving higher levels of consistency. In Section 5, we define a notion of intricacy and use it to characterize tractable cases of deduction. The next three sections contain applications. In Section 9, we generalize the notion of partial consistency to directional consistency. We also generalize the entire framework in the next section. Due to space limitations, detailed discussion and all the proofs have been relegated to [Dalal, 1992b].

2 PRELIMINARIES

We restrict to first-order predicate calculus with equality but without functions symbols (other than constants) and variables [Mendelson, 1964]. Let $\mathcal{L}$ denotes the set of all formulas in this language. Without loss of any generality, we assume that the only logical connectives are $\vee, \wedge$, and $\neg$, and that all formulas are in *negation normal form*, i.e., all negation symbols are in front of the atoms. For technical reasons, we assume a total ordering $\prec$ among all the constants in $\mathcal{L}$. We denote constant symbols by a, b, etc., predicate symbols by p, q, etc., literals (atoms and their negations) by α, β, etc., formulas by ψ, σ, etc., and theories by Γ, Σ, etc. The complement operator $\sim$ on literals is defined as follows: if $\alpha = p(a_1, \ldots, a_k)$ then $\sim\alpha = \neg p(a_1, \ldots, a_k)$, and if $\alpha = \neg p(a_1, \ldots, a_k)$ then $\sim\alpha = p(a_1, \ldots, a_k)$. We assume that there are two special literals, $\mathbf{t}$ (true) and $\mathbf{f}$ (false), that are complements of each other. For any set, A, of literals, $\sim A$ denotes the set containing the complement of each literal in A.

For any theory Γ, we use lits(Γ) to denote the set of all atoms (and their negations) that occur in Γ; facts(Γ) to denote the set of all unit formulas in Γ; and nf(Γ) to denote the set of all non-unit formulas in Γ, i.e., the set (Γ − facts(Γ)). For example, if $\Gamma = \{p(a),\ \neg q(b),\ p(b) \vee \neg q(a)\}$, then lits($\Gamma$) = $\{p(a), \neg p(a), p(b), \neg p(b), q(a), \neg q(a), q(b), \neg q(b)\}$, and facts($\Gamma$) = $\{p(a), \neg q(b)\}$. Following the usual conventions, we denote a singleton set $\{\psi\}$ by ψ, the empty set $\{\}$ by $\emptyset$, and the limit ordinal for natural numbers by ∞.

3 FACT PROPAGATION

We first present a simple way to detect some inconsistent theories. Consider a boolean function BI, called the *basic inconsistency function*, that returns true for a theory iff the set of facts (i.e., unit formulas) in the theory is inconsistent. BI detects some, but not all, logical inconsistencies. For instance, although both the theories $\{p(a), \neg p(a)\}$ and $\{p(a) \vee q(b),\ p(a) \vee \neg q(b),\ \neg p(a) \vee q(b),\ \neg p(a) \vee \neg q(b)\}$ are logically inconsistent, BI can detect the inconsistency in only the former. However, it can be determined in linear time whether BI returns true for any given theory.

We now describe a tractable algorithm FPF, called the *fact propagation function* [Dalal, 1992a; Dalal, 1992b] that simplifies any given logical theory by propagating its literals to other formulas in it. Any new literals that are generated are also propagated until no more new literals can be generated or a basic inconsistency is detected. For example, given the theory $\{p(a),\ \neg p(a) \vee \neg q(a),\ q(a) \vee r(a) \vee s(a)\}$, FPF obtains a logically equivalent theory $\{p(a),\ \neg q(a),\ r(a) \vee s(a)\}$. For theories in conjunctive normal form (CNF), FPF is identical to unit resolution [Chang and Lee, 1973] as well as boolean constraint propagation [McAllester, 1980; McAllester, 1990]. For arbitrary formulae, FPF is more powerful than BCP applied to their CNF form. FPF can also be viewed as a generalization of a linear-time algorithm for testing propositional Horn satisfiability [Dowling and Gallier, 1984].

Given any theory Γ in $\mathcal{L}$, FPF applies the *rewrite rules* of Figure 1, each of which is a logical identity, in any sequence until no more changes are possible. Note that x and y are any constants, $\overline{v}$ and $\overline{w}$ are any sequences of constants, α is any literal, $\psi, \psi_1, \ldots$ are any subformulas, and $[\psi]^{z}_{y}$ denotes the result obtained from ψ by substituting each occurrence of y by z.[2]

For another example, consider the theory Γ =

[2] A theory is viewed as a (possibly infinite) conjunction of all its formulas. Rule 9 is a simplified version of the correct rule in which all the α's are required only to be equivalent under renaming of equivalent constants. Rule 12 uses the total ordering of constants in $\mathcal{L}$.

$\{p(a,b),\ \neg p(c,b), a = c \vee b = c\}$. FPF first obtains $a \neq c$ using Rule 11, and then uses rules 5, 7 and 2 to obtain $b = c$. Assuming $b \prec c$, Rule 12 simplifies the theory further to finally obtain $\text{FPF}(\Gamma) = \{p(a,b),\ \neg p(b,b),\ a \neq b\}$.

By maintaining proper data structures, $\text{FPF}(\Gamma)$ for any theory Γ can be computed in time $O(n(l+1))$, where n is the size of Γ and l is the maximum number of alternations of connectives $\wedge$ and $\vee$ in any formula of Γ. In general, FPF maintains logical equivalence, i.e., for any theory Γ, $\Gamma \equiv \text{FPF}(\Gamma)$. For any theory Γ and set A of literals, we abbreviate $\text{FPF}(\Gamma \cup A)$ by Γ_A. In particular, $\text{FPF}(\Gamma)$ is denoted by $\Gamma_\emptyset$.

4 PARTIAL CONSISTENCY

We now combine BI and FPF to obtain stronger methods to detect more subtle inconsistencies in theories. In particular, we define levels of partial consistency based on the result of assuming and propagating literals in a given theory.

Consider the following logically equivalent theories: $\Delta = \{p(a) \vee q(a),\ p(a) \vee \neg q(a),\ \neg p(a) \vee q(a) \vee r(a)\}$ and $\Omega = \{p(a),\ q(a) \vee r(a)\}$. It can be verified that $\Delta_{\neg p(a)}$ is basic inconsistent, but there is no literal in $\text{nf}(\Omega)$ that causes such basic inconsistency.[3] Thus, in some sense Ω is "more consistent" than Δ. This notion is formalized below:

Definition 1 *Any theory Γ is* (-1)-consistent. *For any natural number k, a theory Γ is* k-consistent *iff* $\neg\text{BI}(\Gamma_\emptyset)$ *and for each literal $\alpha \in \text{lits}(\text{nf}(\Gamma_\emptyset))$, Γ_α is $(k-1)$-consistent. A theory Γ is* ∞-consistent *iff it is k-consistent for all $k < \infty$. The* consistency level *of any theory Γ is the maximum k such that Γ is k-consistent.*

Intuitively, a theory is k-consistent iff propagating any sequence of k assumptions, each of which occurs in the non-facts of the theory after the propagation of earlier assumptions, does not lead to basic inconsistency. In the previous example, while the theory Δ is not even 1-consistent, Ω is ∞-consistent.

Lemma 1 *For any theory Γ and any natural number k, if Γ is k-consistent then it is also $(k-1)$-consistent. A theory is basic inconsistent iff it is not 0-consistent. Any ∞-consistent theory is logically consistent.* ■

Any consistent theory can be transformed into a logically equivalent theory with an arbitrary high level of consistency, by first adding a number of clauses and then doing fact propagation. In the previous example, applying FPF to $\Delta \cup \{p(a)\}$ produces Ω, which is an ∞-consistent theory. Thus, adding $p(a)$ and propagating it changes the consistency level of Δ from 0 to ∞ without changing its logical content. The algorithm

[3]Note that $\text{nf}(\Omega) = \{q(a) \vee r(a)\}$.

```
make-k-consistent(Γ):
{input: any theory Γ;
output: a theory Δ s.t.
   Δ = Δ_∅ ≡ Γ, and
   either BI(Δ) or
    Δ is k-consistent}
begin
 Δ := mconsistent(Γ, k);
 return (Γ ∪ Δ)_∅
end.


mconsistent(Γ, k):
 if k < 0 then return ∅;
 Δ := ∅; Γ := Γ_∅;
 repeat
  if BI(Γ) then return Δ;
  select an α ∈ lits(nf(Γ));
  Δ' := mconsistent(Γ_α, k − 1);
  if BI((Γ_α ∪ Δ')_∅)
   then add ∼α to both Γ and Δ
   else if Δ' ≠ ∅ then
    for each ψ ∈ Δ'
     add ∼α ∨ ψ to Γ and Δ
 until no more changes;
 return Δ
end.
```

Figure 2: An Algorithm for achieving k-Consistency

make-k-consistent, given in Figure 2, performs this transformation for any input theory Γ. Note that for an inconsistent theory, this transformation may produce $\{\mathbf{f}\}$.

For example, consider a theory, $\Delta = \{p(a) \vee q(a),\ p(a) \vee \neg q(a),\ \neg p(a) \vee \neg q(a) \vee r(a),\ \neg p(a) \vee \neg q(a) \vee \neg r(a)\}$. Δ is not 1-consistent, since $\Delta_{\neg p(a)}$ is basic inconsistent. However, **make-1-consistent**(Δ) produces the theory $\{p(a),\ \neg q(a)\}$, that is ∞-consistent. Consider another theory, $\Omega = \{p(a) \vee q(a),\ p(a) \vee \neg q(a),\ r(a) \vee s(a) \vee t(a),\ r(a) \vee s(a) \vee \neg t(a)\}$. **Make-1-consistent**$(\Omega)$ produces a 1-consistent theory $\{p(a),\ r(a) \vee s(a) \vee t(a),\ r(a) \vee s(a) \vee \neg t(a)\}$, while **make-2-consistent**(Ω) produces an ∞-consistent theory $\{p(a),\ r(a) \vee s(a)\}$. Consider the inconsistent theory, $\Sigma = \{p(a) \vee q(a),\ p(a) \vee \neg q(a),\ \neg p(a) \vee q(a),\ \neg p(a) \vee \neg q(a)\}$. It can be verified that **make-1-consistent**(Σ) produces a basic inconsistent theory.

Lemma 2 *For any theory Γ and any natural number k,* **make-k-consistent**(Γ) *always halts and produces a logically equivalent theory Δ such that either* $\text{BI}(\Delta)$ *or Δ is k-consistent.* ■

In [Dalal, 1992b], we show that although there are other algorithms for achieving k-consistency, **make-k-consistent** is special since it makes minimal changes in the input theory. In particular, if $\text{BI}(\Gamma)$ or Γ is k-

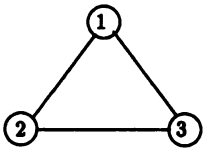


Figure 3: Network for C

$$(1{:}1 \wedge 2{:}3) \vee (1{:}1 \wedge 2{:}4) \vee (1{:}2 \wedge 2{:}3) \vee (1{:}2 \wedge 2{:}4)$$
$$(1{:}1 \wedge 3{:}6) \vee (1{:}2 \wedge 3{:}5) \vee (1{:}2 \wedge 3{:}6)$$
$$(2{:}3 \wedge 3{:}5) \vee (2{:}4 \wedge 3{:}5) \vee (2{:}4 \wedge 3{:}6)$$
$$1{:}1 \vee 1{:}2 \qquad \neg 1{:}1 \vee \neg 1{:}2$$
$$2{:}3 \vee 2{:}4 \qquad \neg 2{:}3 \vee \neg 2{:}4$$
$$3{:}5 \vee 3{:}6 \qquad \neg 3{:}5 \vee \neg 3{:}6$$

Figure 4: $\Delta(C)$

consistent, then **make-k-consistent**$(\Gamma) = \Gamma$. We also show that a straightforward implementation of this algorithm takes $O(m^{k^2} nl)$ time and $O(n + m^k)$ space, where n is the size of Γ, m is the number of distinct symbols in Γ, and l is the maximum number of alternations of connectives $\wedge$ and $\vee$ in any formula of Γ. However, by using some additional data structures, the time complexity can be reduced to $O(m^k nl)$, without any increase in the space complexity. Thus, for a fixed k, this algorithm runs in polynomial time.

5 TRACTABLE DEDUCTION

The algorithm **make-k-consistent** of the previous section transforms any theory into a logically equivalent theory that is either basic inconsistent or k-consistent. As we saw in some examples there, sometimes the result may be ∞-consistent even for a small value of k. In such cases it is trivial to determine whether the given theory is consistent. Different theories may require different values of k to achieve this condition.

Definition 2 *The* intricacy *of a theory, Γ, is the minimum k such that* **make-k-consistent**(Γ) *is either basic inconsistent or ∞-consistent. The* intricacy *of a collection, S, of theories is the minimum of the following two: (1) the maximum intricacy of all consistent theories in S, and (2) the maximum intricacy of all inconsistent theories in S.*

In the previous example, the intricacy[4] of Δ, Ω, and Σ are 1, 2, and 1, respectively. In some sense, the intricacy of a theory is a measure of the difficulty of determining its consistency. The next theorem shows that the same holds for collections of theories.

Theorem 1 *If the intricacy of a collection, S, of theories in $\mathcal{L}$ is finite, then the consistency problem for S can be solved in polynomial time.* ■

Thus, intricacy is a parameter which can be used to identify tractable cases for determining consistency. [Dalal, 1992b] presents an algorithm, which is a variant of **make-k-consistent**, that determines consistency for such theories in polynomial time. Note that the above theorem gives only a sufficient condition, that may not be necessary, i.e., there may exist a tractable collection of theories with unbounded intricacy. However, [Dalal, 1992c] shows that the two most well-known tractable cases of propositional satisfiability — Horn clauses and 2-SAT — have finite intricacy.

[4] In [Dalal, 1992c], the term "cryptness" was used for the notion of intricacy.

6 CONSTRAINT SATISFACTION

A constraint satisfaction problem [Mackworth, 1987] is specified by a finite set, X, of variables $\{x_1, \ldots, x_n\}$ and a set of constraints on subsets of these variables limiting the values they can take. A solution is an assignment of a value to each variable such that all the constraints are satisfied. As is customary in the CSP literature, we restrict our attention to those CSPs where all the constraints are either unary or binary and are explicitly provided as sets of tuples. It is well-known that the general problem of determining whether a CSP has a solution is intractable.

The network representing a CSP problem is a graph whose nodes represent variables and unary constraints, and arcs represent binary constraints. Current efforts [Dechter and Pearl, 1988] to characterize tractable CSPs rely either on the topology of the underlying constraint network or the semantics of the constraints [Deville and Hentenryck, 1991]. In this section, we present an overview of how intricacy has been used to develop a new and much more powerful characterization of tractable CSPs [Dalal, 1992c]. In addition to the topology of the constraint network, this characterization also uses the semantics and the syntactic forms of various constraints in the network.

Any CSP problem C can be directly translated into a theory $\Delta(C)$ in $\mathcal{L}$ such that the size of $\Delta(C)$ is polynomial in the size of C, and that C has a solution iff $\Delta(C)$ is logically consistent. For example, consider a CSP problem, C, with the network shown in Figure 3, and where the unary constraints for x_1, x_2, and x_3 are $\{1,2\}$, $\{3,4\}$, and $\{5,6\}$ respectively; and the binary constraints for the pairs $\{x_1, x_2\}$, $\{x_1, x_3\}$, and $\{x_2, x_3\}$ are $\{(1,3),(1,4),(2,3),(2,4)\}$, $\{(1,6),(2,5),(2,6)\}$, and $\{(3,5),(4,5),(4,6)\}$ respectively. The formulas of the mapping, $\Delta(C)$, are shown in Figure 4, where an atom $i{:}j$ intuitively denotes that variable x_i is assigned value j. Note that $\Delta(C)$ is not

in CNF. Although there are other ways to map CSPs to propositional satisfiability [de Kleer, 1989], many of them lead to an exponential increase in size.

Notions of k-consistency and strong k-consistency have been defined for CSP problems [Freuder, 1978]. We show in [Dalal, 1992b] that, in a certain sense, our notion of k-consistency of $\Delta(C)$ is stronger than either of these notions for C. In particular, we prove the following lemma:

Lemma 3 *If a CSP problem, C, with n variables is strong n-consistent then $\Delta(C)$ is ∞-consistent. For any $k > 2$, there is a CSP problem, C, that is not strong k-consistent, but $\Delta(C)$ is ∞-consistent. For any $k > 1$, there is a CSP problem, C, that is strong k-consistent, but $\Delta(C)$ is not even 1-consistent.* ■

For example, the network, C, of Figure 3 is not 3-consistent, since the consistent partial assignment $\{1: 1, 2:3\}$ can't be extended to a consistent assigument of x_3. However, $\Delta(C)$ is ∞-consistent, since 2:4 holds in $\Delta(C)_{1:1}$, and 1:2 holds in $\Delta(C)_{2:3}$. Intuitively, the inconsistencies are detected and corrected earlier in $\Delta(C)$.

The next theorem shows that intricacy of $\Delta(C)$ is bounded by each of the topological parameters that are currently used for identifying tractable CSPs.

Theorem 2 *For any CSP problem, C, the intricacy of $\Delta(C)$ is less than each of the following parameters of the network for C: induced-width, the size of the largest non-separable component, the minimum depth of a DFS spanning tree, and the size of the smallest cycle-cutset.* ■

It follows that the worst-case time complexity of determining whether a solution exists is polynomially bounded by any of these parameters, a result that's already known. This theorem is significant, considering the following last line in the survey [Dechter, 1991]: "We conclude, therefore, that ... (induced-width) ... is the most informative graph-parameter of all, and it can be regarded, therefore, as an intrinsic measure of the worst-case complexity of any constraint network.".

Intricacy can also be used to identify tractable classes of CSPs based on the semantics of the constraints. A functional constraint [Deville and Hentenryck, 1991] is defined to be a binary constraint such that for each value of one variable, at most one value of the other variable is allowed. A CSP is called functional if its constraint network contains a spanning tree in which each edge corresponds to a functional constraint. For any fixed number k, it is easy to construct a functional CSP for which each of the topological parameters listed above is greater than k. Since none of these parameters can be bounded, topological criteria alone fail to identify functional CSPs as a tractable class. However, it follows from the next theorem [Dalal, 1992b] that it can be determined in polynomial time whether any functional network has a solution.

Theorem 3 *The intricacy of $\Delta(C)$ is 1 for any functional CSP problem C.*

We conclude, therefore, that our characterization of tractable CSPs based on intricacy is a significant improvement over any such characterization based either on the topology of the underlying network or the semantics of the constraints alone.

7 QUERYING DATABASES WITH INCOMPLETE INFORMATION

Recent results [Vardi, 1986; Abiteboul *et al.*, 1987] have shown that allowing representation of even very simple kinds of incomplete information in relational databases makes it intractable (CoNP-Complete) to query them. Although there have been some attempts (c.f. [Abiteboul *et al.*, 1987; Imielinski and Vadaparty, 1989]) to identify tractable cases of querying, they are applicable only to databases with very restricted kinds of incomplete information. For example, the only kind of incompleteness allowed in [Imielinski and Vadaparty, 1989] is the presence of certain variables, each of which can take a value from a pre-specified set of constants. In this section, we define a very general model of relational databases with incomplete information, and use intricacy to obtain tractable cases of querying. We pay no penalty for the generality of our model, since our tractability results subsume those known previously.

The Data Model: Consider a first-order logical language $\mathcal{L}_d$ with equality but without function symbols (other than constants). A *database* DB consists of two parts — an *intensional database* IDB and an *extensional database* EDB. An IDB is a theory in $\mathcal{L}_d$ that does not contain any existential quantifier. An EDB is a theory in $\mathcal{L}_d$ that contains only ground formulas. Although we denote a DB by a tuple $\langle IDB, EDB\rangle$, logically it is the union of the two theories. The *domain* D of a database is the set of all constants that occur in it.

This data model is more powerful than Datalog [Ullman, 1988] and can represent various kinds of incomplete information — simple, restricted and conditional null values [Codd, 1979; Imielinski and Lipski, 1984], uniform disjunctions [Imielinski and Vadaparty, 1989; Borgida and Etherington, 1989], conditional tuples [Abiteboul *et al.*, 1987], disjunctive rules [Gallaire *et al.*, 1984], etc. It can also represent restricted forms of explicit negative information, and integrity constraints of various kinds. However, this model cannot represent information which involves functions, or which involves an existential quantifier in the scope of a universal quantifier. Although the closed-world assumption [Reiter, 1984] is ignored in this section, [Dalal, 1992b] shows that it can be easily incorporated.

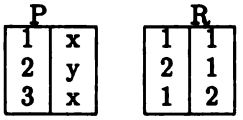


Figure 5: An OR-database

$P(1,x)$	$Q(1,1)$	$x=1 \vee x=2$
$P(2,y)$	$Q(2,1)$	$y=1 \vee y=2$
$P(3,x)$	$Q(1,2)$	$\neg P(X,Y) \vee \neg Q(Y,X)$

Figure 6: $Tr(\mathrm{DB}, Q, ())$

A *query* Q is an expression of the form $(\overline{x}).\psi(\overline{x})$, where ψ is a formula that does not contain any universal quantifier, and $\overline{x}$ is a sequence of all free variables that occur in ψ. The answer Q(DB) is the set $\{\overline{a} \in D^{|\overline{x}|} : \mathrm{DB} \models \psi(\overline{a})\}$. The *data complexity* of a query Q for an intensional database IDB is the complexity of their *graph*, which is defined as:

$$Gr(\mathrm{IDB}, Q) = \{(\mathrm{EDB}, \overline{a}) \mid \overline{a} \in Q(\langle \mathrm{IDB}, \mathrm{EDB} \rangle)\}$$

Since IDB and Q are kept fixed, this measures the complexity of query evaluation in terms of the size of EDB.

Given any database DB, query Q, and a tuple $\overline{a}$, [Dalal, 1992b] presents a polynomial time algorithm that obtains a theory $Tr(\mathrm{DB}, Q, \overline{a})$ in $\mathcal{L}$, which is unsatisfiable iff $\overline{a} \in Q(\mathrm{DB})$. It follows that the problem of querying databases with incomplete information can be polynomially reduced to the deduction problem in $\mathcal{L}$. Thus, we can use the notion of intricacy to identify tractable cases of querying.

For any query $Q = (\overline{x}).\psi(\overline{x})$ and any intensional database IDB, consider the following set:

$$\Gamma(\mathrm{IDB}, Q) = \{Tr(\langle \mathrm{IDB}, \mathrm{EDB} \rangle, Q, \overline{a}) \mid \overline{a} \in D^{|\overline{x}|} \text{ and EDB is any extensional database}\}$$

Theorem 4 *The data complexity of query Q for an intensional database IDB is in polynomial time if the intricacy of* $\Gamma(\mathrm{IDB}, Q)$ *is finite.*

Consider the following three special kinds of incompleteness and queries:

1. The DB is in CNF, any clause contains at most two literals, equality predicate is not used (though inequality is allowed), and the query is a conjunctive formula with at most two literals.
2. The DB contains marked nulls (i.e., null values that can be shared), and conditions that involves only inequality. The query could be any conjunctive formula.
3. Similar to the case 1 above, except that equality is also allowed.

Querying is tractable in the first two cases, since the intricacy is finite [Dalal, 1992b]. However, intricacy is not finite in the third case, for which the data complexity is indeed CoNP-Complete.

OR-databases: OR-databases extend relational data model by allowing explicit representation of disjunctive information [Imielinski and Vadaparty, 1989]. A tuple may contain OR-objects — variables each of which is associated with a set of constants, called its *domain*. Each OR-database corresponds to potentially many relational databases, obtained by substituting each OR-object by some value in its domain. A query, which is any existentially quantified conjunctive formula with no free variables, is true for an OR-database iff it is true in all the relational databases associated with it.

Three kinds of OR-databases are considered in [Imielinski and Vadaparty, 1989]: Type-I (all OR-objects are distinct), Type-II (an OR-object can be repeated only in the same column), and Type-III (no restrictions). Databases are further grouped by associating a *typing function* which specifies the columns that can have OR-objects. For any kind of OR databases, and any typing function, tractability of any query can be determined using a syntactic marking property. For example, a query is tractable with respect to Type-II databases iff it is acyclic. Similar results are proved for Type-I and Type-III databases.

Since OR-databases can be represented in our data model (where IDB is always empty), the translation $Tr(\mathrm{DB}, Q, ())$ can be defined for any database DB and query Q. Consider the OR-database of Figure 5, where both the OR-objects x and y have the same domain $\{1,2\}$, and the query $\exists x \exists y(P(x,y) \wedge R(y,x))$. Their translation contains the formulas given in Figure 6, where the last schema represents all formulas obtained by substituting each of X and Y by a value in $\{1,2\}$ in all possible ways. In this particular case, the query is true since $Tr(\mathrm{DB}, Q, ())$ is inconsistent. As an abbreviation, we define the intricacy of a query Q to be the intricacy of the set $\Gamma(\emptyset, Q)$ of logical theories. The following theorem shows that all the tractability results presented in [Imielinski and Vadaparty, 1989] could be explained using the notion of intricacy:

Theorem 5 *The intricacy of any acyclic query for Type-II OR-databases, or any strongly acyclic query for Type-I OR-databases, or any simple query for Type-III OR-databases is bounded by the number of OR-arguments in the query.*

A *binary query* contains at most two literals. Using the notion of intricacy, we now obtain a new tractability

result for OR-databases of Type I, where no sharing of OR-objects is allowed:

Theorem 6 *The intricacy of any binary query for Type-I OR-databases, in which the domain of each OR-object contains at most two constants and each relation has at most one OR-column, is 1.*

Thus, such databases can also be queried in polynomial time. In [Dalal, 1992b] we consider some other data models that allow explicit representation of incomplete information (for instance, see [Abiteboul *et al.*, 1987]) and obtain tractable cases of querying them.

8 QUERYING DISJUNCTIVE LOGIC PROGRAMS

For the purpose of this section, we restrict to propositional calculus. A *disjunctive logic program* consists of a finite set of *program clauses* of the form

$$\alpha \leftarrow \beta_1, \ldots, \beta_k \qquad (k \geq 0)$$

where $\alpha, \beta_1, \ldots, \beta_k$ are disjunctions of atoms (called wfds). α is called the *head* and $\beta_1, \ldots, \beta_k$ is called the *body* of the program clause. Similarly, a *goal clause* (or a *query*) is an expression of the form

$$\leftarrow \beta_1, \ldots, \beta_k \qquad (k \geq 0)$$

A query Q is true in a program P iff Q is a logical consequence of P.

It is known that the complexity of querying disjunctive logic programs is CoNP-Complete [Lobo, 1990]. In this section, we use the notion of intricacy to identify cases of tractable querying. Since we are only interested in positive wfds that are logically entailed by a disjunctive logic program, the definition of partial-consistency is modified so that α is required to be a negative literal. The same change is made in the algorithm **make-k-consistent**. The definition of intricacy, etc. remains the same.

As in the case of definite logic programs, where each disjunction contains exactly one atom, a declarative semantics can be provides to capture the intended meaning of a disjunctive logic program [Minker and Rajasekar, 1990; Lobo, 1990; Dalal, 1992b]. The *Herbrand base* $\mathrm{HB}(P)$ of any program P is the set of all atoms that occur in P. The *extended Herbrand base* $\mathrm{EHB}(P)$ is the set of all wfds that can be formed using the atoms in $\mathrm{HB}(P)$, i.e., $\mathrm{EHB}(P) = 2^{HB(P)}$. A *state* S is a subset of $\mathrm{EHB}(P)$ that is *closed wrt supersets*, i.e.,

$$\forall \alpha, \beta \in \mathrm{EHB}(P), \quad (\alpha \in S \wedge \alpha \subseteq \beta) \rightarrow \beta \in S$$

A state S is a *model-state* of a program clause $\alpha \leftarrow \beta_1, \ldots, \beta_k$ iff $\forall \delta_1, \ldots, \delta_k \in S$, $\alpha \cup (\delta_1 - \beta_1) \cup \ldots \cup (\delta_k - \beta_k) \in S$. A state is a *model-state* of a disjunctive program iff it is a model-state of each clause in the program.

Intuitively, the notion of a state (or model-state) is analogous to the notion of a Herbrand interpretation (model) for definite programs. For instance, the intersection of model-states produces a model-state (model-state intersection property). Thus, the intersection of all model-states of a program P is also a model-state, called the *least model-state*, of P. Following the usual convention, we denote this model-state by M_P. The wfds in M_P are precisely those that are logical consequences of the program. Thus, M_P can be regarded as the natural interpretation of the program. It follows that a query is true for a program iff each wfd in the query is also in the least model-state of the program.

For example, consider the program $P = \{a \vee b \leftarrow,\ c \leftarrow a,\ c \leftarrow b\}$ containing only three clauses. The Herbrand base of P is $\{a, b, c\}$, whose power set is the extended Herbrand base. P has four model-states:

$$\begin{aligned} M_1 &= \{c, a \vee b, a \vee c, b \vee c, a \vee b \vee c\} \\ M_2 &= \{a, c, a \vee b, a \vee c, b \vee c, a \vee b \vee c\} \\ M_3 &= \{b, c, a \vee b, a \vee c, b \vee c, a \vee b \vee c\} \\ M_4 &= \{a, b, c, a \vee b, a \vee c, b \vee c, a \vee b \vee c\} \end{aligned}$$

Note that M_1, which is the least model-state of P, captures the intended meaning of the program.

The least model-state of any program P can be alternatively characterized as the least fixed-point of a mapping $T_P : 2^{EHB(P)} \rightarrow 2^{EHB(P)}$ that is defined as follows:

$$\begin{aligned} T_P(S) = \{\alpha' \in \mathrm{EHB}(P) : {} & \alpha \leftarrow \beta_1, \ldots, \beta_k \text{ is a program} \\ & \text{clause in } P, \alpha \cup (\delta_1 - \beta_1) \cup (\delta_k - \beta_k) \\ & \subseteq \alpha', \text{ and } \{\delta_1, \ldots, \delta_k\} \subseteq S\} \end{aligned}$$

Intuitively, this mapping extends the mapping T_P for definite programs [van Emden and Kowalski, 1976] to disjunctive programs by explicitly enforcing the logical dependencies among the wfds. [Dalal, 1992b] shows that T_P is continuous (and monotonic) for any disjunctive program P, and hence $T_P \uparrow \omega$ is its least fixed-point. It also shows that this least fixed-point is identical to the least model-state of the program, which is its intended meaning.

For any program P, let $\mathrm{head}(P)$ denote the set of heads of all program clauses whose bodies are empty. The next lemma shows that for any ∞-consistent program P, its least model-state can be generated using $\mathrm{head}(P)$. It then follows that achieving a consistency level of intricacy is sufficient to generate the least model-state of a program.

Lemma 4 *For any disjunctive program P*

1. *if P is ∞-consistent then $M_P = [\mathrm{head}(P)]$; and*
2. *if the intricacy of P is k then $M_P = [\mathrm{head}(\textbf{make-}k\textbf{-consistent}(P))]$.*

Since any query can be answered using the least model-state of the program, we obtain the next theorem which shows that querying programs of finite intricacy is tractable.

Theorem 7 *If the intricacy of a set S of programs is finite, then the programs in S can be queried in polynomial time.*

Thus, intricacy is a parameter which can be used to identify tractable cases of querying. We illustrate its use by showing two simple examples. Consider a definite program P. Propagating its facts, without making any assumptions, produces the theory $P_\emptyset$, which is ∞-consistent. Thus, the intricacy of P is 0. Note that head($P_\emptyset$) is also the least Herbrand model of P. It follows that any definite program can be queried in linear time. Consider another program P, such that each program clause in P contains at most two atoms. As shown in the example above, such a P may not be a definite program. In [Dalal, 1992b], we show that intricacy of any such program is 1, and hence can also be queried in polynomial time.

In [Dalal, 1992b], we use intricacy to define a notion of approximating the least model-state of a program. In contrast to the approximations in [Lobo, 1990], our approximations are always sound.

9 DIRECTIONAL CONSISTENCY

In this section we improve the results of Sections 4 and 5 by relaxing the notion of k-consistency. In particular, we require that the literals are selected in some fixed order. For simplicity of presentation, we restrict to propositional calculus.

Definition 3 *For any theory Γ, an ordering, o, of symbols is a sequence in which each symbol in Γ occurs exactly once. For any literal p, the ordering o_p is obtained from o by removing p and all the symbols that occur before it.*

For example, $o = [P, Q, R, S]$ is an ordering for the theory $\{\neg P \vee Q \vee R,\ R \vee \neg S\}$. For this ordering, $o_Q = o_{\neg Q} = [R, S]$.[5] Directional consistency can be defined as follows:

Definition 4 *Let o be any ordering for a theory Γ. Any Γ is (-1)-consistent wrt o. For any natural number k, Γ is k-consistent wrt o iff $\neg$bi(Γ_ϕ) and for each literal $p \in$ lits(nf(Γ_ϕ)) such that p is in o, Γ_p is $(k-1)$-consistent wrt o_p.* ■

Intuitively, directional consistency does not allow the selection of literals in an order that violates the given ordering. In the above example, literal P can't be selected after selecting literal $\neg R$. For any k, a theory may not be k-consistent, but it may be k-consistent wrt some particular ordering. Consider the theory $\Delta = \{\neg P \vee Q,\ \neg P \vee \neg Q \vee \neg R,\ \neg R \vee P \vee Q\}$. Since $\Delta_{\{R, \neg Q\}}$ is basic inconsistent, Δ_R is not 1-consistent, and thus Δ is not 2-consistent. However, it can be verified that Δ is 2-consistent wrt the ordering $[P, Q, R]$. Intuitively, this ordering precludes selecting the literal $\neg Q$ in Δ_R, while $\neg R$ already holds in $\Delta_{\neg Q}$. However, the following lemma holds:

Lemma 5 *For any k, a theory is k-consistent iff it is k-consistent wrt every ordering.* ■

In [Dalal, 1992b], we show that the notions of ∞-consistent, consistency level, and intricacy of a theory can be analogously defined wrt a given ordering. The directional intricacy of a theory is the minimum intricacy wrt any ordering. Directional consistency can be achieved by modifying the algorithm **mconsistent** of Figure 2. The modified algorithm keeps track of the literals selected so far, and does not allow them to violate the given ordering. We also show that directional counterparts of Lemmas 1 and 2 hold. Theorem 1 can also be further strengthened by introducing a new notion: an *ordering function*, of, is a polynomial function from the set of theories to the set of orderings. Intuitively, for any theory, it generates an ordering of symbols for which it is easier to achieve either ∞-consistency or basic inconsistency. The intricacy of a set of theories is now defined wrt a particular ordering function, of, so that the intricacy of each theory, Γ, in the set is considered wrt to the ordering of(Γ). The directional version of Theorem 1 is as follows:

Theorem 8 *For any collection, S, of propositional theories, if the intricacy of S wrt to some ordering function is finite, then consistency problem for S can be solved in polynomial time.* ■

Since the directional counterpart of Lemma 3 holds for constraint satisfaction problems, it follows that this notion of directional consistency of $\Delta(C)$ is stronger than the notion of adaptive consistency [Dechter and Pearl, 1988].

10 A GENERAL THEORY

Until now we have been using specific basic inconsistency and fact propagation functions. In this section, we generalize our results by using any such functions which satisfies certain properties.

A *basic inconsistency function* is any polynomial time boolean function on the set of all theories. A basic inconsistency function, bi, is *sound* iff for any theory Γ, if bi(Γ) is true then Γ is unsatisfiable. bi is *complete wrt facts* iff for any theory Γ, if facts(Γ) is unsatisfiable then bi(Γ) is true. A theory, Γ, is said to be *basic inconsistent* iff bi(Γ) holds. It can be verified that BI is a basic inconsistency function which is both sound and complete wrt facts.

[5] We abuse the notation a little bit by ignoring the distinction between a symbol and its negation, and say that $\neg R$ is in the sequence $[P, Q, R, S]$.

A *fact propagation function* is any polynomial time function on the set of all theories. A fact propagation function, fpf, is *sound* iff for any theory Γ, $\Gamma \models \text{fpf}(\Gamma)$. fpf is *lossless* iff any theory Γ, $\text{fpf}(\Gamma) \models \Gamma$. fpf is *separable* iff for any theory Γ, the sets of symbols occurring in facts(fpf(Γ)) and nf(fpf(Γ)) are disjoint. fpf is *monotonic* iff for any theory Γ, either *bi*(fpf(Γ)) or facts(Γ) $\subseteq$ facts(fpf(Γ)). A fact propagation function, fpf, is said to be *complete wrt clauses* iff for any theory Γ containing a clause C, if p is a literal in C then $\Gamma_{\sim p}$ is either basic inconsistent or there is clause D in it such that p does not occur in D, and all the literals of D occur in C. For example, if Γ contains the clause $P \vee Q \vee R$, then $\Gamma_{\neg P}$ should be either basic inconsistent or should contain Q, R, or $Q \vee R$. It can be verified that FPF is sound, lossless, separable, monotonic, and complete wrt clauses.

The results in this paper hold for any basic inconsistency function that is sound and complete wrt facts, and any fact propagation function that is sound, lossless, separable, monotonic, and complete wrt clauses.

11 OTHER RELATED RESEARCH

Gallo and Scutella [1988] develop a hierarchy of classes of instances of SAT, such that for each class, satisfiability can be determined in polynomial time. Our characterization is stronger than theirs since we can identify all the tractable classes in their hierarchy. In particular, the kth class of their hierarchy has intricacy $(k+1)$. They also restrict the formulas to clauses, and none of their classes contain all the 2CNF formulas.

Ben-Eliyahu and Dechter [1991] introduce the notion of a "primal constraint graph" for a propositional theory — the nodes of the graph correspond to the propositional symbols, and the arcs connect any two nodes whose symbols appear in the same formula. They determine the tractability of a class of theories by the topological parameters, like the induced-width, of such graphs. As we have shown in Section 6, this characterization is weaker than ours since it does not take into account the semantics of the formulas.

Our approach of obtaining higher consistency levels of a theory has some similarities to the process of identifying "nogood" sets in an ATMS [de Kleer, 1986]. However, we are not aware of any extension of that process that achieves results similar to ours.

12 CONCLUSIONS

We defined notions of partial consistency and intricacy for logical theories, and used them to identify many cases of tractable deduction. To the best of our knowledge, our tractability results are the most powerful that have ever been obtained in the areas of constraint satisfaction problems, databases with incomplete information and disjunctive logic programming. In [Dalal, 1992b], we use the techniques presented here to obtain tractability results for several other problems like propositional planning, default reasoning, and subsumption in concept languages.

The techniques presented in this paper have been introduced only as theoretical tools to study complexity; we do not suggest that they be used for actual problem solving. Thus, finding efficient algorithms for the tractable cases identified in this paper remains a useful direction for future research.

Acknowledgements

I would like to thank Alex Borgida, David Etherington, Bart Selman, Hari Hampapuram, Kumar Vadaparty and Vipul Kashyap for helpful discussions and comments on the earlier drafts of this paper. I also wish to thank AT&T Bell Laboratories for partial support during this research.

References

(Abiteboul *et al.*, 1987) Abiteboul, S.; Kanellakis, P.; and Grahne, G. 1987. On the representation and querying of sets of possible worlds. In *Proceedings ACM SIGMOD International Conference on Management of Data*, San Francisco, California. 34–48.

(Ben-Eliyahu and Dechter, 1991) Ben-Eliyahu, R. and Dechter, R. 1991. Default logic, propositional logic and constraints. In *Proceedings Ninth National Conference on Artificial Intelligence (AAAI-91)*. 379–385.

(Borgida and Etherington, 1989) Borgida, A. and Etherington, D.W. 1989. Hierarchical knowledge bases and efficient disjunctive reasoning. In Brachman, R.J.; Levesque, H.J.; and Reiter, R., editors 1989, *Proceedings First International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann. 33–43.

(Brachman and Levesque, 1984) Brachman, R.J. and Levesque, H.J. 1984. The tractability of subsumption in frame based description languages. In *Proceedings National Conference on Artificial Intelligence (AAAI-84)*. 34–37.

(Chang and Lee, 1973) Chang, C. and Lee, R.C. 1973. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, London.

(Codd, 1970) Codd, E.F. 1970. A relational model for large shared data banks. *Communications of ACM* 13(6):377–387.

(Codd, 1979) Codd, E.F. 1979. Extending the relational database to capture more meaning. *ACM Transactions of Database Systems* 4(4):397–434.

(Cook, 1971) Cook, S.A. 1971. The complexity of theorem proving procedures. In *Proceedings Third Annual ACM Symposium on the Theory of Computing.* 151–158.

(Dalal, 1992a) Dalal, M. 1992a. Efficient propositional constraint propagation. In *Proceedings Tenth National Conference on Artificial Intelligence (AAAI-92)*, San Jose, California. American Association for Artificial Intelligence. 409–414.

(Dalal, 1992b) Dalal, M. 1992b. Tractable deduction in knowledge representation systems. Ph.D. thesis (in preparation).

(Dalal, 1992c) Dalal, M. 1992c. Tractable instances of some hard deduction problems. In *Proceedings Tenth European Conference on Artificial Intelligence (ECAI '92)*, Vienna, Austria.

(de Kleer, 1986) de Kleer, J. 1986. An assumption-based truth maintenance system. *Artificial Intelligence* 28:127–162.

(de Kleer, 1989) de Kleer, J. 1989. A comparison of ATMS and CSP techniques. In *Proceedings Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89).* 290–296.

(Dechter and Pearl, 1988) Dechter, R. and Pearl, J. 1988. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence* 34(1):1–38.

(Dechter, 1991) Dechter, R. 1991. Constraint satisfaction. In Shapiro, S.C., editor 1991, *Encyclopedia of AI (2nd Edition).* John Wiley and Sons.

(Deville and Hentenryck, 1991) Deville, Y. and Hentenryck, P.V. 1991. An efficient arc consistency algorithm for a class of CSP problems. In *Proceedings Twelveth International Joint Conference on Artificial Intelligence (IJCAI-91).* 325–330.

(Dowling and Gallier, 1984) Dowling, W.F. and Gallier, J.H. 1984. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming* 1(3):267–284.

(Doyle and Patil, 1991) Doyle, J. and Patil, R. 1991. Two theses of knowledge representation: language restrictions, taxanomic classification, and the utility of representation services. *Artificial Intelligence* 48(3):261–297.

(Freuder, 1978) Freuder, E.C. 1978. Synthesizing constraint expressions. *Communications of the ACM* 21(11):958–966.

(Gallaire *et al.*, 1984) Gallaire, H.; Minker, J.; and Nicolas, J.M. 1984. Logic and databases: A deductive approach. *ACM Computing surveys* 16(2):153–185.

(Gallo and Scutella, 1988) Gallo, G. and Scutella, M.G. 1988. Polynomially solvable satisfiability problems. *Information Processing Letters* 29:221–227.

(Garey and Johnson, 1979) Garey, M. and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, W. H., NY.

(Imielinski and Lipski, 1984) Imielinski, T. and Lipski, W. 1984. Incomplete information in relational databases. *Journal of the ACM* 31(4):761–791.

(Imielinski and Vadaparty, 1989) Imielinski, T. and Vadaparty, K. 1989. Complexity of query processing in databases with OR-objects. In *Proceedings ACM Symposium on Principles of Database Systems*, Philedelphia, Pennsylvania. 51–65. Also in a recent (unpublished) draft.

(Levesque and Brachman, 1985) Levesque, H.J. and Brachman, R.J. 1985. A fundamental tradeoff in knowledge representation and reasoning (revised version). In Brachman, R.J. and Levesque, H.J., editors 1985, *Readings in Knowledge Representation.* Morgan Kaufmann, Los Altos, California. 41–70.

(Lobo, 1990) Lobo, J. 1990. *Semantics for Normal Disjunctive Logic Programs.* Ph.D. Dissertation, Department of Computer Science, University of Maryland, College Park.

(Mackworth, 1987) Mackworth, A.K. 1987. Constraint satisfaction. In Shapiro, S.C., editor 1987, *Encyclopedia of AI.* John Wiley and Sons. 205–211.

(McAllester, 1980) McAllester, D. 1980. An outlook on truth maintenance. Memo 551, MIT AI Lab.

(McAllester, 1990) McAllester, D. 1990. Truth maintenance. In *Proceedings Eight National Conference on Artificial Intelligence (AAAI-90).* 1109–1116.

(Mendelson, 1964) Mendelson, E. 1964. *Introduction to Mathematical Logic.* Van Nostrand, Princeton, N.J.

(Minker and Rajasekar, 1990) Minker, J. and Rajasekar, A. 1990. A fixpoint semantics for disjunctive logic programs. *Journal of Logic Programming* 9(1):45–74.

(Reiter, 1984) Reiter, R. 1984. Towards a logical reconstruction of relational database theory. In Brodie, M.L.; Mylopoulos, J.; and Schmidt, J.W., editors 1984, *On Conceptual Modelling.* Springer-Verlag, New York. chapter 8, 191–233.

(Ullman, 1988) Ullman, J.D. 1988. *Principles of Database and Knowledge-Base Systems*, volume 1. computer science press, Rockville, MD.

(van Emden and Kowalski, 1976) van Emden, M.H. and Kowalski, R.A. 1976. The semantics of predicate logic as a programming language. *Journal of the ACM* 23(4):733–742.

(Vardi, 1986) Vardi, M. Y. 1986. Querying logical databases. *Journal of Computer and System Sciences* 33(2):142–160.

New Results on Local Inference Relations

Robert Givan and David McAllester
MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge Mass. 02139

Abstract

We consider the concept of a *local* set of inference rules. A local rule set can be automatically transformed into a rule set for which bottom up evaluation terminates in polynomial time. The local rule set transformation gives polynomial time evaluation strategies for a large variety of rule sets that can not be given terminating evaluation strategies by any other known automatic technique. This paper discusses three new results. First, it is shown that every polynomial time predicate can be defined by an (unstratified) local rule set. Second, a new machine recognizable subclass of the local rule sets is identified. Finally we show that locality, as a property of rule sets, is undecidable in general.

1 INTRODUCTION

Under what conditions does a given set of inference rules define a computationally tractable inference relation? This is a syntactic question about syntactic inference rules. There are a variety of motivations for identifying tractable inference relations. First, tractable inference relations sometimes provide decision procedures for semantic theories. For example, the equational inference rules of reflexivity, symmetry, transitivity, and substitutivity define a tractable inference relation that yields a decision procedure for the entailment relation between sets of ground equations [Kozen, 1977], [Shostak, 1978]. Another example is the set of equational Horn clauses valid in lattice theory. As a special case of the results in this paper one can show (automatically) that validity of a lattice theoretic Horn clause is decidable in cubic time.

Deductive data bases provide a second motivation for studying tractable inference relations. A deductive data base is designed to answer queries using simple inference rules as well as a set of declared data base facts. The inference rules in a deductive data base usually define a tractable inference relation. The inference rules are usually of a special form known as a datalog program. A datalog program is a set of first order Horn clauses that do not contain function symbols. Any datalog program defines a tractable inference relation [Ullman, 1988], [Ullman, 1989]. Recently there has been interest in generalizing the inference rules used in deductive databases beyond the special case of datalog programs. In the general case, where function symbols are allowed in Horn clause inference rules, a set of inference rules can be viewed as a Prolog program. Considerable work has been done on "bottom up" evaluation strategies for these programs and source to source transformations that make such bottom up evaluation strategies more efficient [Naughton and Ramakrishnan, 1991] [Bry, 1990]. The work presented here on local inference relations can be viewed as an extension of these optimization techniques. For example, locality testing provides an automatic source to source transformation on the inference rules for equality (symmetry, reflexivity, transitive, and substitution) that allows them to be completely evaluated in a bottom-up fashion in cubic time. We do not know of any other automatic transformation on inference rules that provides a terminating evaluation strategy for this rule set.

A third motivation for the study of tractable inference relations is the role that such relations can play in improving the efficiency of search. Many practical search algorithms use some form of incomplete inference to prune nodes in the search tree [Knuth, 1975], [Mackworth, 1977], [Pearl and Korf, 1987]. Incomplete inference also plays an important role in pruning search in constraint logic programming [Jaffar and Lassez, 1987], [van Hentenryck, 1989], [McAllester and Siskind, 1991]. Tractable inference relations can also be used to define a notion of "obvious inference" which can then be used in "Socratic" proof verification systems which require proofs to be reduced to obvious steps [McAllester, 1989], [Givan *et al.*, 1991].

As mentioned above, inference rules are syntactically similar to first order Horn clauses. In fact, most infer-

ence rules can be syntactically represented by a Horn clause in sorted first order logic. If R is a set of Horn clauses, Σ is a set of ground atomic formulas, and Φ is a ground atomic formula, then we write $\Sigma \vdash_R \Phi$ if $\Sigma \cup R \vdash \Phi$ in first order logic. We write $\vdash_R$ rather than $\models_R$ because we think of R as a set of syntactic inference rules and $\vdash_R$ as the inference relation generated by those rules. Throughout this paper we use the term "rule set" as a synonym for "set of Horn clauses". Technically this phrase refers to a finite set of Horn clauses. We give nontrivial conditions on R which ensure that the inference relation $\vdash_R$ is polynomial time decidable.

As noted above, a rule set R that does not contain any function symbols is called a datalog program. It is well known that the inference relation defined by a datalog program is polynomial time decidable. Vardi and Immerman independently proved, in essence, that datalog programs provide a characterization of the complexity class P — any polynomial time predicate on finite data bases can be written as a datalog program provided that one is given a successor relation that defines a total order on the domain elements [Vardi, 1982], [Immerman, 1986], [Papadimitriou, 1985].

Although datalog programs provide an interesting class of polynomial time inference relations, the class of tractable rule sets is much larger than the class of datalog programs. First of all, one can generalize the concept of a datalog program to the concept of a *superficial* rule set. We call a set of Horn clauses superficial if any term that appears in the conclusion of a clause also appears in some antecedent of that clause. A superficial rule set has the property that forward chaining inference does not introduce new terms. Superficial rule sets provide a different characterization of the complexity class P. While datalog programs can encode any polynomial time predicate on finite data bases, superficial rule sets can encode any polynomial time predicate on first order terms. Let $\mathcal{P}$ be a predicate on first order terms constructed from a finite signature. We define the DAG size of a first order term t to be the number of distinct terms that appear as subexpressions of t.[1] It is possible to show that if $\mathcal{P}$ can be computed in polynomial time in sum of the the DAG size of its arguments then $\mathcal{P}$ can be represented by a superficial rule set. More specifically, we prove here that for any such predicate $\mathcal{P}$ on k first order terms there exists a superficial rule set R such that $\mathcal{P}(t_1, t_2, \ldots t_k)$ if and only if $\texttt{Input}(t_1, t_2, \ldots t_k) \vdash_R \texttt{Accept}$ where Input is a predicate symbol and Accept is a distinguished proposition symbol. The characterization of P in terms of superficial rule sets differs from Immerman's characterization of P in terms of datalog programs in two ways. First, the result is stated in terms of predicates on terms rather than predicates on databases. Second, unlike the datalog characterization, no separate total order on domain elements is required.

Superficial rule sets are a special case of the more general class of *local* rule sets [McAllester, 1990]. A set R of Horn clauses is local if whenever $\Sigma \vdash_R \Phi$ there exists a proof of Φ from Σ such that every term in the proof is mentioned in Σ or Φ. If R is local then $\vdash_R$ is polynomial time decidable. All superficial rule sets are local but many local rule sets are not superficial. The set of the four inference rules for equality is local but not superficial. The local inference relations provide a third characterization of the complexity class P. Let $\mathcal{P}$ be a predicate on first order terms constructed from a finite signature. If $\mathcal{P}$ can be computed in polynomial time in the sum of the DAG size of its arguments then there exists a local rule set R such that for any terms $t_1, t_2, \ldots t_k$ we have that $\mathcal{P}(t_1, t_2, \ldots t_k)$ if and only if $\vdash_R P(t_1, t_2, \ldots t_k)$ where P is a predicate symbol representing $\mathcal{P}$. Note that no superficial rule set can have this property because forward chaining inference from a superficial rule set can not introduce new terms. We find the characterization of polynomial time predicates in terms of local rule sets to be particularly pleasing because it yields a direct mapping from semantic predicates to predicates used in the inference rules.

Unlike superficiality, locality can be difficult to recognize. The set of four inference rules for equality is local but the proof of this fact is nontrivial. Fortunately, there are large classes of mechanically recognizable local rule sets. A notion of a bounded local rule set is defined in [McAllester, 1990] and a procedure is given which will automatically recognize the locality of any bounded local rule set. The set of the four basic rules for equality is bounded local. As another example of a bounded local rule set we give the following rules for reasoning about a monotone operator from sets to sets.

$$x \leq x$$
$$x \leq y,\ y \leq z \Rightarrow x \leq z$$
$$x \leq y \Rightarrow f(x) \leq f(y)$$

Let R_f be this set of inference rules for a monotone operator.

There is a simple source to source transformation on any local rule set that converts the rule set to a superficial rule set without losing completeness. For example consider the above rules for a monotone operator. We can transform these rules so that they can only derive information about terms explicitly mentioned in the query. To do this we introduce another predicate symbol M (with the intuitive meaning "mentioned"). The rules can be rewritten as follows.

$$\texttt{M}(f(x)) \Rightarrow \texttt{M}(x)$$

[1] The DAG size of a term is the size of the Directed Acyclic Graph representation of the term.

$$x \leq y \Rightarrow \mathtt{M}(x)$$
$$x \leq y \Rightarrow \mathtt{M}(y)$$
$$\mathtt{M}(x) \Rightarrow x \leq x$$
$$\mathtt{M}(x), \mathtt{M}(y), \mathtt{M}(z), x \leq y, y \leq z \Rightarrow x \leq z$$
$$\mathtt{M}(f(x)), \mathtt{M}((f(y)), x \leq y \Rightarrow f(x) \leq f(y)$$

Let this transformed rule set be R'_f. Note that R'_f is superficial and hence bottom up (forward chaining) evaluation must terminate in polynomial time.[2] Then to determine if $\Sigma \vdash_{R_f} t \leq u$ we determine, by bottom up evaluation, whether $\{\mathtt{M}(t), \mathtt{M}(u)\} \cup \Sigma \vdash_{R'_f} t \leq u$. An analogous transformation applies to any local rule set.

A variety of other bounded local rule sets are given [McAllester, 1990]. As an example of a rule set that is local but not bounded local we give the following rules for reasoning about a lattice.

$$x \leq x$$
$$x \leq y, y \leq z \Rightarrow x \leq z$$
$$x \leq x \vee y$$
$$y \leq x \vee y$$
$$x \leq z, y \leq z \Rightarrow x \vee y \leq z$$
$$x \wedge y \leq x$$
$$x \wedge y \leq y$$
$$z \leq x, z \leq y \Rightarrow z \leq x \wedge y$$

These rules remain local when the above monotonicity rule is added. With or without the monotonicity rule, the rule set is not bounded local.

In this paper we construct another machine-recognizable subclass of the local rule sets which we call *inductively local* rule sets. All of the bounded local rule sets given in [McAllester, 1990] are also inductively local. The procedure for recognizing inductively local rule sets has been implemented and has been used to determine that the above rule set is inductively local. Hence the inference relation defined by the rules is polynomial time decidable. Since these rules are complete for lattices this result implies that validity for lattice theoretic Horn clauses is polynomial time decidable.

We been able to show that there are bounded local rule sets which are not inductively local, although our examples are somewhat artificial. We have not found any natural examples of local rule sets that fail to be inductively local. Inductively local rule sets provide a variety of mechanically recognizable polynomial time inference relations.

[2]For this rule set bottom up evaluation can be run to completion in cubic time.

In this paper we also settle an open question from our previous analysis [McAllester, 1990] and show that locality as a general property of rule sets is undecidable. Hence the optimization of logic programs based on the recognition of locality is necessarily a somewhat heuristic process.

2 BASIC TERMINOLOGY

In this section we give more precise definitions of the concepts discussed in the introduction.

> **Definition:** A Horn clause is a first order formula of the form $\Psi_1 \wedge \Psi_2 \wedge \cdots \wedge \Psi_n \Rightarrow \Phi$ where Φ and the Ψ_i are atomic formulas. For any set of Horn clauses R, any finite set Σ of ground terms, and any ground atomic formula Φ, we write $\Sigma \vdash_R \Phi$ whenever $\Sigma \cup U(R) \vdash \Phi$ in first order logic where $U(R)$ is the set of universal closures of Horn clauses in R.

There are a variety of inference relations defined in this paper. For any inference relation $\vdash$ and sets of ground formulas Σ and Γ we write $\Sigma \vdash \Gamma$ if $\Sigma \vdash \Psi$ for each Ψ in Γ.

The inference relation $\vdash_R$ can be given a more direct syntactic characterization. This syntactic characterization is more useful in determining locality.

> **Definition:** A *derivation* of Φ from Σ using rule set R is a sequence of ground atomic formulas $\Psi_1, \Psi_2, \ldots \Psi_n$ such that Ψ_n is Φ and for each Ψ_i there exists a Horn clause $\Theta_1 \wedge \Theta_2 \wedge \ldots \wedge \Theta_k \Rightarrow \Psi'$ in R and a ground substitution σ such that $\sigma[\Psi'] = \Psi_i$ and each formula of the form $\sigma[\Theta_i]$ is either a member of Σ or a formula appearing in earlier in the derivation.
>
> **Lemma:** $\Sigma \vdash_R \Phi$ if and only if there exists a derivation of Φ from Σ using the rule set R.

The following restricted inference relation plays an important role in the analysis of locality.

> **Definition:** We write $\Sigma \vdash\!\!\!\!\!\!\cdot_R \Phi$ if there exists a derivation of Φ from Σ such that every term appearing in the derivation appears as a subexpression of Φ or as a subexpression of some formula in Σ.
>
> **Lemma:** For any finite rule set R the inference relation $\vdash\!\!\!\!\!\!\cdot_R$ is polynomial time decidable.

Proof: Let n be the number of terms that appear as subexpressions of Φ or a formula in Σ. If P is a predicate of k arguments that appears in the inference

rules R then there are at most n^k formulas of the form $P(s_1, \ldots s_k)$ such that $\Sigma \vdash_R P(s_1, \ldots s_k)$. Since R is finite there is some maximum arity over all the predicate symbols that appear in R. The total number of formulas that can be derived under the restrictions in the definition of $\vdash_R$ is order n^k where k is the maximum arity of the predicates in R. ■

Clearly, if $\Sigma \vdash_R \Phi$ then $\Sigma \vdash_R \Phi$. But the converse does not hold in general. By definition, if the converse holds then R is local.

> **Definition([McAllester, 1990]):** The rule set R is *local* if the restricted inference relation $\vdash_R$ is the same as the unrestricted relation $\vdash_R$.

Clearly, if R is local then $\vdash_R$ is polynomial time decidable.

3 CHARACTERIZING P WITH SUPERFICIAL RULES

In this section we consider predicates on first order terms that are computable in polynomial time. The results stated require a somewhat careful definition of a polynomial time predicate on first order terms.

> **Definition:** A *polynomial time predicate on terms* is a predicate $\mathcal{P}$ on one or more first order terms which can be computed in polynomial time in the sum of the DAG sizes of its arguments.

> **Superficial Rule Set Representation Theorem:** If $\mathcal{P}$ is a polynomial time predicate on first order terms then there exists a superficial rule set R such that for any first order terms $t_1, \ldots, t_n$, we have that $\mathcal{P}$ is true on arguments $t_1, \ldots, t_n$ if and only if $\texttt{INPUT}(t_1, \ldots, t_n) \vdash_R \texttt{ACCEPT}$.

As an example consider the *Acyclic* predicate on directed graphs — the predicate which is true of a directed graph if that graph has no cycles. It is well known that acyclicity is a polynomial time property of directed graphs. This property has a simple definition using superficial rules with one level of stratification — if a graph is not cyclic then it is acyclic. The above theorem implies that the acyclicity predicate can be defined by superficial rules without any stratification. The unstratified rule set for acyclicity is somewhat complex and rather than give it here we sketch a proof of the above general theorem. The proof is rather technical, and casual readers are advised to skip to the next section.

We only consider predicates of one argument. The proof for predicates with of higher arity is similar. Let $\mathcal{P}$ be a one argument polynomial time predicate on terms, i.e., a predicate on terms such that one can determine in polynomial time in the DAG size of a term t whether or not $\mathcal{P}(t)$ holds. We construct a data base that represents the term t. For each subterm s of t we introduce a data base individual c_s, i.e., a new constant symbol unique to the term s. We have assumed that the predicate $\mathcal{P}$ is only defined on terms constructed from a fixed finite signature, i.e., a fixed finite set of constant and function symbols. We will consider constants to be functions of no arguments. For each function symbol f of n arguments in this finite signature we introduce a database relation Q_f of $n+1$ arguments, i.e., Q_f is a $n+1$-ary predicate symbol. Now for any term t we define Σ_t to be the set of ground formulas of the form $Q_f(c_{f(s_1 \ldots s_n)}, c_{s_1}, \ldots, c_{s_n})$ where $f(s_1, \ldots, s_n)$ is a subterm of t (possibly equal to t). The set Σ_t should be viewed as a data base with individuals c_s and relations Q_f. Let Γ be a set of formulas of the form $S(c_s, c_u)$ where s and u are subterms of t such that S represents a successor relation on the individuals of Σ_t, i.e., there exists a bijection ρ from the individuals of Σ_t to consecutive integers such that $S(s, u)$ is in Γ if and only if $\rho(u) = \rho(s)+1$. The result of Immerman and Vardi [Immerman, 1986], [Vardi, 1982] implies that for any polynomial time property of the set $\Sigma_t \cup \Gamma$ there exists a datalog program R such that $\Sigma_t \cup \Gamma$ has the given property if and only if $\Sigma_t \cup \Gamma \vdash_R \texttt{ACCEPT}$. Since the term t can be easily recovered from the set Σ_t, there must exist a datalog program R such that $\Sigma_t \cup \Gamma \vdash_R \texttt{ACCEPT}$ if and only if $\mathcal{P}(t)$. We can assume without loss of generality that no rule in R can derive new formulas involving the data base predicates Q_f.

We now add to the rule set R superficial rules that construct analogues of the formulas in Σ_t and Γ. First we define a mentioned predicate $\texttt{M}$ such that $\texttt{M}(s)$ is provable if and only if s is a subterm of t.

$$\texttt{INPUT}(t) \Rightarrow \texttt{M}(t)$$

$$\texttt{M}(f(x_1, \ldots, x_n)) \Rightarrow \texttt{M}x_i$$

The second rule is a schema for all rules of this form where f is one of the finite number of function symbols in the signature and x_i is one of the variables $x_1, \ldots, x_n$. Now we give rules that simulate the formula set Σ_t.

$$\texttt{M}(f(x_1, \ldots, x_n)) \Rightarrow Q_f(f(x_1, \ldots, x_n), x_1, \ldots, x_n)$$

Now we write rules that simulate the formula set Γ, i.e., that define a successor relation on the terms in t. We start by defining a simple subterm predicate Su such that $Su(s, u)$ is provable if s and u are subterms of t such that s is a subterm of u.

$$\texttt{M}(x) \Rightarrow Su(x, x)$$

$$\texttt{M}(f(x_1, \ldots, x_n)), Su(y, x_i) \Rightarrow Su(y, f(x_1, \ldots, x_n))$$

Next we define a "not equal" predicate NE such that $NE(s, u)$ is provable if and only if s and u are distinct subterms of the input t.

$$\mathtt{M}(f(x_1, \ldots, x_n)), \mathtt{M}(g(y_1, \ldots, y_m)) \Rightarrow NE(f(x_1, \ldots, x_n), g(y_1, \ldots, y_m))$$

$$\left.\begin{array}{r}\mathtt{M}(f(x_1, \ldots, x_i, \ldots x_n)),\\ f(x_1, \ldots, y_i, \ldots x_n),\\ NE(x_i, y_i)\end{array}\right\} \Rightarrow NE(f(x_1, \ldots, x_i, \ldots x_n), f(x_1, \ldots, y_i, \ldots x_n))$$

In the first rule f and g must be distinct function symbols and in the second rule x_i and y_i occur at the same argument position and all other arguments to f are the same in both terms. Next we define a "not in" predicate NI such that $NI(s, u)$ if s is not a subterm of u. We only give the rules for constants and functions of two arguments. The rules for functions of other numbers of arguments are similar. In the following rule c must be a constant.

$$NE(s, c) \Rightarrow NI(s, c)$$

$$NE(z, f(x,y)), NI(z, x), NI(z, y) \Rightarrow NI(z, f(x, y))$$

Now for any subterm s of the input we simultaneously define a three place "walk" relation $W(s, u, w)$ and a binary "last" relation $L(s, u)$. $W(s, u\ w)$ will be provable if s and u are subterms of w and u is the successor of s in a left to right preorder traversal of the subterms of w with elimination of later duplicates. $L(s, u)$ will be provable if s is the last term of the left to right preorder traversal of the subterms of u, again with elimination of later duplicates. In these definitions, we also use the auxiliary three place relation W', which can be viewed as "try to conclude W, checking for duplicates". Once again, c must be a constant.

$$\mathtt{M}(f(x, y)) \Rightarrow W(f(x, y), x, f(x, y))$$

$$\mathtt{M}(f(x, y)), W(u, v, x) \Rightarrow W(u, v, f(x, y))$$

$$\mathtt{M}(f(x, y)), NI(y, x), L(s, x) \Rightarrow W(s, y, f(x, y))$$

$$\mathtt{M}(f(x, y)), W(u, v, y) \Rightarrow W'(u, v, f(x, y))$$

$$W'(u, v, f(x, y)), NI(u, x), NI(v, x) \Rightarrow W(u, v, f(x, y))$$

$$W'(u, v, f(x, y)), W'(v, w, f(x, y)), Su(v, x) \Rightarrow W'(u, w, f(x, y))$$

$$\Rightarrow L(c, c)$$

$$\mathtt{M}(f(x, y)), L(ylast, y), NI(ylast, x) \Rightarrow L(ylast, f(x, y))$$

$$\mathtt{M}(f(x, y)), Su(y, x), L(xlast, x) \Rightarrow L(xlast, f(x, y))$$

$$L(ylast, y), Su(ylast, x), NI(y, x), W'(flast, ylast, f(x, y)), W(u, flast, f(x, y)) \Rightarrow L(flast, f(x, y))$$

Finally we define the successor predicate S.

$$\mathtt{INPUT}(z), W(x, y, z) \Rightarrow S(x, y)$$

Let R' be the the datalog program R plus all of the above superficial rules. We now have that $\Sigma_t \cup \Gamma \vdash_R \mathtt{ACCEPT}$ if and only if $\mathtt{INPUT}(t) \vdash_{R'} \mathtt{ACCEPT}$ and the proof is complete.

4 CHARACTERIZING P WITH LOCAL RULES

Using the theorem of the previous section one can provide a somewhat different characterization of the complexity class P in terms of local rule sets.

> **Local Rule Set Representation Theorem:** If $\mathcal{P}$ is a polynomial time predicate on first order terms then there exists a local rule set R such that for any first order terms $t_1, \ldots, t_n$, we have that $\mathcal{P}$ is true on arguments $t_1, \ldots, t_n$ if and only if $\vdash_R P(t_1 \ldots t_n)$ where P is a predicate symbol representing $\mathcal{P}$.

Before giving a proof of this theorem we give a simple example of a local rule set for a polynomial time problem. Any context free language can be recognized in cubic time. This fact is easily proven by giving a translation of grammars into local rule sets. We represent a string of words using a constant symbol for each word and the binary function CONS to construct terms that represent lists of words. For each nonterminal symbol A of the grammar we introduce a predicate symbol P_A of two arguments where $P_A(x\ y)$ will indicate that x and y are strings of words and that y is the result of removing a prefix of x that parses as category A. For each production $A \rightarrow c$ where c is a terminal symbol we construct a rule with no antecedents and the conclusion $P_A(\mathtt{CONS}(c, x)\ x)$. For each grammar production $A \rightarrow BC$ we have the following inference rule.

$$P_B(x\ y) \wedge P_C(y\ z) \rightarrow P_A(x\ z)$$

Finally, we let P be a monadic predicate which is true of strings generated by the distinguished start nonterminal S of the grammar and add the following rule.

$$P_S(x\ \mathtt{NIL}) \Rightarrow P(x)$$

Let R be this set of inference rules. R is a local rule set, although the proof of locality is not entirely trivial. The rule set R also has the property that $\vdash_R P(x)$ if and only if x is a string in the language generated by the given grammar. General methods for analyzing the order of running time of local rule sets can be used to immediately give that these clauses can be run to completion in order n^3 time where n is the length of the input string.[3] We have implemented a compiler for converting local rule sets to efficient inference procedures. This compiler can be used to automatically generate a polynomial time parser from the above inference rules.

We now prove the above theorem for local inference relations from the preceding theorem for superficial rule sets. By the preceding theorem there must exist a superficial rule set R such that for any first order terms $t_1, t_2, \ldots t_k$ we have that $\mathcal{P}(t_1, t_2, \ldots t_n)$ if and only if $\texttt{INPUT}(t_1, t_2, \ldots t_k) \vdash_R \texttt{ACCEPT}$ where INPUT is a predicate symbol and ACCEPT is a distinguished proposition symbol. For each predicate symbol Q of m arguments appearing in R let Q' be a new predicate symbol of $k+m$ arguments. We define the rule set R' to be the rule set containing the following clauses.

- $\texttt{Input}'(x_1, \ldots x_k, x_1, \ldots x_k)$
- All clauses of the form

$$Q'_1(x_1, \ldots x_k, t_{1,1}, \ldots t_{1,m_1}) \wedge \cdots \wedge Q'_n(x_1, \ldots x_k, t_{n,1}, \ldots t_{n,m_n}) \Rightarrow W'(x_1, \ldots x_k, s_1, \ldots s_j)$$

  where the clause $Q_1(t_{1,1}, \ldots t_{1,m_1}) \wedge \cdots \wedge Q_n(t_{n,1}, \ldots t_{n,m_n}) \Rightarrow W(s_1, \ldots s_j)$ is in R.
- The clause $\texttt{Accept}'(x_1, \ldots x_k) \Rightarrow P(x_1, \ldots x_k)$.

Given the above definition we can easily show that $\vdash_{R'} Q'(t_1, \ldots t_k, s_1, \ldots s_m)$ if and only if $\texttt{Input}(t_1, \ldots t_k) \vdash_R Q(s_1, \ldots s_m)$. Therefore, it follows that $\texttt{Input}(t_1, \ldots t_k) \vdash_R \texttt{Accept}$ if and only if $\vdash_{R'} P(t_1, \ldots t_k)$. It remains only to show that R' is local. Suppose that $\Sigma \vdash_{R'} \Phi$. We must show that $\Sigma \vdash_{R'} \Phi$. Let $t_1, \ldots t_k$ be the first k arguments in Φ. Every inference based on R' involves formulas which all have the same first k arguments. Given that $\Sigma \vdash_{R'} \Phi$ we must have that $\Sigma' \vdash_{R'} \Phi$ where Σ' is the set of formulas in Σ that have $t_1, \ldots t_k$ as their first k arguments. Let Σ'' be and Φ' be the result of replacing each formula $Q'(t_1, \ldots t_k, s_1, \ldots, s_m)$ by $Q(s_1, \ldots, s_m)$. Since $\Sigma' \vdash_{R'} \Phi$ we must have $\{\texttt{Input}(t_1, \ldots t_k)\} \cup \Sigma'' \vdash_R \Phi'$. But since R is superficial this implies that every term in the derivation underlying $\{\texttt{Input}(t_1, \ldots t_k)\} \cup \Sigma'' \vdash_R \Phi'$ either appears in some t_i or appears in Σ''. This implies that every term in the derivation appears in either Σ' or Φ. This implies $\Sigma \vdash_{R'} \Phi$.

[3] An analysis of the order of running time for decision procedures for local inference relations is given in [McAllester, 1990].

5 ANOTHER CHARACTERIZATION OF LOCALITY

In this section we give an alternate characterization of locality. This characterization of locality plays an important role in both the definition of bounded local rule sets given in [McAllester, 1990] and in the notion of inductively local rule sets given here.

Definition: A *bounding set* is a set Υ of ground terms such that every subterm of a member of Υ is also a member of Υ.

Definition: A ground atomic formula Ψ is called a *label formula* of a bounding set Υ if every term in Ψ is a member of Υ.

Definition: For any bounding set Υ, we define the inference relation $\vdash_{R,\Upsilon}$ to be such that $\Sigma \vdash_{R,\Upsilon} \Phi$ if and only if there exists a derivation of Φ from Σ such that every formula in the derivation is a label formula of the term set Υ.

We have that $\Sigma \vdash_R \Phi$ if and only if $\Sigma \vdash_{R,\Upsilon} \Phi$ where Υ is the set of all terms appearing as subexpressions of Φ or formulas in Σ. The inference relation $\vdash_{R,\Upsilon}$ can be used to give another characterization of locality. Suppose that R is not local. In this case there must exist some Σ and Φ such that $\Sigma \not\vdash_R \Phi$ but $\Sigma \vdash_R \Phi$. Let Υ be the set of terms that appear in Σ and Φ. We must have $\Sigma \not\vdash_{R,\Upsilon} \Phi$. However, since $\Sigma \vdash_R \Phi$ we must have $\Sigma \vdash_{R,\Upsilon'} \Phi$ for some finite superset Υ' of Υ. Consider "growing" the bounding set one term at a time, starting with the terms that appear in Σ and Φ.

Definition: A *one step extension* of a bounding set Υ is a ground term α that is not in Υ but such that every proper subterm of α is a member of Υ.

Definition: A feedback event for R consists of a finite set Σ of ground formulas, a ground formula Φ, a bounding set Υ containing all terms that appear in Σ and Φ, and a one step extension α of Υ such that $\Sigma \vdash_{R,\Upsilon\cup\{\alpha\}} \Phi$ but $\Sigma \not\vdash_{R,\Upsilon} \Phi$.

By abuse of notation, a feedback event will be written as $\Sigma \vdash_{R,\Upsilon\cup\{\alpha\}} \Phi$.

Lemma([McAllester, 1990]): *R is local if and only if there are no feedback events for R.*

Proof: First note that if R has a feedback event then R is not local — if $\Sigma \vdash_{R,\Upsilon\cup\{\alpha\}} \Phi$ then $\Sigma \vdash_R \Phi$ but if $\Sigma \not\vdash_{R,\Upsilon} \Phi$ then $\Sigma \not\vdash_R \Phi$. Conversely suppose

that R is not local. In there case there is some Σ and Φ such that $\Sigma \not\vdash_R \Phi$ but $\Sigma \vdash_{R,\Upsilon} \Phi$ for some finite Υ. By considering a least such Υ one can show that a feedback event exists for R. ∎

The concepts of bounded locality and inductive locality both involve the concept of a feedback event. We can define bounded locality by first defining $C_R(\Sigma, \Upsilon)$ to be the set of of formulas Ψ such that $\Sigma \vdash_{R,\Upsilon} \Psi$. R is bounded local if it is local and there exists a natural number k such that whenever $\Sigma \vdash_{R,\Upsilon\cup\{\alpha\}} \Psi$ there exists a derivation of Ψ from $C_R(\Sigma, \Upsilon)$ such that every term in the derivation is a member of $\Upsilon \cup \{\alpha\}$ and such that the derivation is no longer than k. As mentioned above, the set of the four basic inference rules for equality is bounded local and there exists a procedure which can recognize the locality of any bounded local rule set. The definition of inductive locality is somewhat more involved and is given in the next section.

6 INDUCTIVE LOCALITY

To define inductive locality we first define the notion of a feedback template. A feedback template represents a set of potential feedback events. We also define a backward chaining process which generates feedback templates from a rule set R. We show that if there exists a feedback event for R then such an event will be found by this backchaining process. Furthermore, we define an "inductive" termination condition on the backchaining process and show that if the backchaining process achieves inductive termination then R is local.

Throughout this section we let R be a fixed but arbitrary set of Horn clauses. The inference relation $\vdash_{R,\Upsilon}$ will be written as $\vdash_\Upsilon$ with the understanding that R is an implicit parameter of the relation.

We define feedback templates as ground objects — they contain only ground first order terms and formulas. The process for generating feedback templates is defined as a ground process — it only deals with ground instances of clauses in R. The ground process can be "lifted" using a lifting transformation. Since lifting is largely mechanical for arbitrary ground procedures [McAllester and Siskind, 1991], the lifting operation is only discussed briefly here.

> **Definition:** A *feedback template* consists of a set of ground atomic formulas Σ, a multiset of ground atomic formulas Γ, a ground atomic formula Φ, a bounding set Υ, and a one step extension α of Υ such that Φ and every formula in Σ is a label formula of Υ, every formula in Γ is a label formula of $\Upsilon\cup\{\alpha\}$ that contains α, and such that $\Sigma \cup \Gamma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$.

By abuse of notation a feedback template will be written as $\Sigma, \Gamma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$. Γ is a multiset of ground atomic formulas, each of which is a label formula of $\Upsilon \cup \{\alpha\}$ containing α, and such that the union of Σ and Γ allow the derivation of Φ relative to the bounding set $\Upsilon \cup \{\alpha\}$. A feedback template is a potential feedback event in the sense that an extension of Σ that allows a derivation of the formulas in Γ may result in a feedback event. The requirement that Γ be a multiset is needed for the induction lemma given below. Feedback templates for R can be constructed by backward chaining.

> **Procedure for Generating a Template for R:**
>
> 1. Let $\Psi_1 \wedge \Psi_2 \wedge \cdots \wedge \Psi_n \Rightarrow \Phi$ be a ground instance of a clause in R.
> 2. Let α be a term that appears in the clause but does not appear in the conclusion Φ and does not appear as a proper subterm of any other term in the clause.
> 3. Let Υ be a bounding set that does not contain α but does contain every term in the clause other than α.
> 4. Let Σ be the set of antecedents Ψ_i which do not contain α.
> 5. Let Γ be the set of antecedents Ψ_i which do contain α.
> 6. Return the feedback template $\Sigma, \Gamma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$.

We let $\mathcal{T}_0[R]$ to be the set of all feedback templates that can be derived from R by an application of the above procedure. We leave it to the reader to verify that $\mathcal{T}_0[R]$ is a set of feedback templates. Now consider a feedback template $\Sigma, \Gamma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$. We can construct a new template by backward chaining from $\Sigma, \Gamma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ using the following procedure.

> **Procedure for Backchaining from $\Sigma, \Gamma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$**
>
> 1. Let Θ be a member of Γ
> 2. Let $\Psi_1 \wedge \Psi_2 \wedge \cdots \wedge \Psi_n \Rightarrow \Theta$ be a ground instance of a clause in R that has Θ as its conclusion and such that each Ψ_i is a label formula of $\Upsilon \cup \{\alpha\}$.
> 3. Let Σ' be Σ plus all antecedents Ψ_i which do not contain α.
> 4. Let Γ' be Γ minus Θ plus all antecedents Ψ_i which do contain α.
> 5. Return the template $\Sigma', \Gamma' \vdash_{\Upsilon\cup\{\alpha\}} \Phi$.

In step 4 of the above procedure, Γ' is constructed using multiset operations. For example, if the multiset Γ contains two occurrences of Θ, then "Γ minus Θ" contains one occurrence of Θ. We need Γ to be a multiset in order to guarantee that backchaining operations commute in the proof of the induction lemma below—in particular, we will use the fact that if a sequence of backchaining operations remove an element

Θ of Γ at some point, then there exists a permutation of that sequence of backchaining operations producing the same resulting template, but that removes Θ first.

For any set $\mathcal{T}$ of feedback templates we define $\mathcal{B}[\mathcal{T}]$ to be $\mathcal{T}$ plus all templates that can be derived from an element of $\mathcal{T}$ by an application of the above backchaining procedure. It is important to keep in mind that by definition $\mathcal{B}[\mathcal{T}]$ contains $\mathcal{T}$. We let $\mathcal{B}^n[\mathcal{T}]$ be $\mathcal{B}[\mathcal{B}[\cdots\mathcal{B}[\mathcal{T}]]]$ with n applications of $\mathcal{B}$.

Definition: A feedback template is called *critical* if Γ is empty.

If $\Sigma,\emptyset \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ is a critical template then $\Sigma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$. If $\Sigma \not\vdash_{\Upsilon} \Phi$ then $\Sigma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ is a feedback event. By abuse of notation, a critical template $\Sigma,\emptyset \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ such that $\Sigma \not\vdash_{\Upsilon} \Phi$ will be called a feedback event. The following lemma provides the motivation for the definition of a feedback template and the backchaining process.

Lemma: There exists a feedback event for R if and only if there exists a j such that $\mathcal{B}^j[\mathcal{T}_0[R]]$ contains a feedback event.

Proof: To prove the above lemma suppose that there exists a feedback event for R. Let $\Sigma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ be a minimal feedback event for R, i.e., a feedback event for R which minimizes the length of the derivation of Φ from Σ under the bounding set $\Upsilon \cup \{\alpha\}$. The fact that this feedback event is minimal implies that every formula in the derivation other than Φ contains α. To see this suppose that Θ is a formula in the derivation other than Φ that does not involve α. We then have $\Sigma \vdash_{\Upsilon\cup\{\alpha\}} \Theta$ and $\Sigma \cup \{\Theta\} \vdash_{\Upsilon\cup\{\alpha\}} \Phi$. One of these two must be a feedback event — otherwise we would have $\Sigma \vdash_{\Upsilon} \Phi$. But if one of these is a feedback event then it involves a smaller derivation than $\Sigma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ and this contradicts the assumption that $\Sigma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ is minimal. Since every formula other than Φ in the derivation underlying $\Sigma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ contains α, the template $\Sigma,\emptyset \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ can be derived by backchaining. ■

The above lemma implies that if the rule set is not local then backchaining will uncover a feedback event. However, we are primarily interested in those cases where the rule set is local. If the backchaining process is to establish locality then we must find a termination condition which guarantees locality. Let $\mathcal{T}$ be a set of feedback templates. In practice $\mathcal{T}$ can be taken to be $\mathcal{B}^j[\mathcal{T}_0[R]]$ for some finite j. We define a "self-justification" property for sets of feedback templates and prove that if $\mathcal{T}$ is self-justifying then there is no n such that $\mathcal{B}^n[\mathcal{T}]$ contain a feedback event. In defining the self-justification property we treat each template in $\mathcal{T}$ as an independent induction hypothesis. If each template can be "justified" using the set of templates as induction hypotheses, then the set $\mathcal{T}$ is self-justifying.

Definition: We write $\Sigma,\Gamma \vdash_{\mathcal{T},\Upsilon} \Phi$ if $\mathcal{T}$ contains templates

$$\Sigma_1,\Gamma_1 \vdash_{\Upsilon\cup\{\alpha\}} \Psi_1$$
$$\Sigma_2,\Gamma_2 \vdash_{\Upsilon\cup\{\alpha\}} \Psi_2$$
$$\vdots$$
$$\Sigma_k,\Gamma_k \vdash_{\Upsilon\cup\{\alpha\}} \Psi_k$$

where each Σ_i is a subset of Σ, each Γ_i is a subset of Γ and $\Sigma\cup\{\Psi_1, \Psi_2, \ldots \Psi_k\} \vdash_{\Upsilon} \Phi$.

Definition: $\mathcal{T}$ is said to *justify* a template $\Sigma,\Gamma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ if there exists a $\Theta \in \Gamma$ such that for each template $\Sigma',\Gamma' \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ generated by backchaining from $\Sigma,\Gamma \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ by selecting Θ at step 1 of the backchaining procedure we have $\Sigma',\Gamma' \vdash_{\mathcal{T},\Upsilon} \Phi$.

Definition: The set $\mathcal{T}$ is called *self-justifying* if every member of $\mathcal{T}$ is either critical or justified by $\mathcal{T}$, and $\mathcal{T}$ does not contain any feedback events.

Induction Theorem: If $\mathcal{T}$ is self-justifying then no set of the form $\mathcal{B}^n[\mathcal{T}]$ contains a feedback event.

Proof: We must show that for every critical template $\Sigma,\emptyset \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ in $\mathcal{B}^n[\mathcal{T}]$ we have that $\Sigma \vdash_{\Upsilon} \Phi$. The proof is by induction on n. Consider a critical template $\Sigma,\emptyset \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ in $\mathcal{B}^n[\mathcal{T}]$ and assume the theorem for all critical templates in $\mathcal{B}^j[\mathcal{T}]$ for j less than n. The critical template $\Sigma,\emptyset \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ must be derived by backchaining from some template $\Sigma',\Gamma' \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ in $\mathcal{T}$. Note that Σ' must be a subset of Σ. If Γ' is empty then Σ' equals Σ and $\Sigma \vdash_{\Upsilon} \Phi$ because $\mathcal{T}$ is assumed not to contain any feedback events. If Γ' is not empty then, since $\mathcal{T}$ is self justifying, there must exist some Θ in Γ' such that for each template $\Sigma'',\Gamma'' \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ derived from $\Sigma',\Gamma' \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ by backchaining on Θ we have $\Sigma'',\Gamma'' \vdash_{\mathcal{T},\Upsilon} \Phi$. We noted above that backchaining operations commute. By the commutativity of backchaining steps there exists a backchaining sequence from $\Sigma',\Gamma' \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ to $\Sigma,\emptyset \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ such that the first step in that sequence is a backchaining step on Θ. Let $\Sigma'',\Gamma'' \vdash_{\Upsilon\cup\{\alpha\}} \Phi$ be the template that results from this backchaining step from $\Sigma',\Gamma' \vdash_{\Upsilon\cup\{\alpha\}} \Phi$. Note that Σ'' is a subset of Σ. We must now have $\Sigma'',\Gamma'' \vdash_{\mathcal{T},\Upsilon} \Phi$. By definition, $\mathcal{T}$ must contain templates

$$\Sigma_1,\Gamma_1 \vdash_{\Upsilon\cup\{\alpha\}} \Psi_1$$
$$\Sigma_2,\Gamma_2 \vdash_{\Upsilon\cup\{\alpha\}} \Psi_2$$
$$\vdots$$
$$\Sigma_k,\Gamma_k \vdash_{\Upsilon\cup\{\alpha\}} \Psi_k$$

such that each Σ_i is a subset of Σ'', each Γ_i is a subset of Γ'', and $\Sigma'' \cup \{\Psi_1, \Psi_2, \ldots \Psi_k\} \vdash_T \Phi$. Note that each Σ_i is a subset of Σ. Since Γ_i is a subset of Γ'' there must be a sequence of *fewer than n* backchaining steps that leads from $\Sigma_i, \Gamma_i \vdash_{T \cup \{\alpha\}} \Psi_i$ to a critical template $\Sigma_i', \emptyset \vdash_{T \cup \{\alpha\}} \Psi_i$. Note that Σ_i' is a subset of Σ. This critical template is a member of $\mathcal{B}^j[T]$ for j less than n so we have $\Sigma_i' \vdash_T \Psi_i$ and thus $\Sigma \vdash_T \Psi_i$. But if $\Sigma \vdash_T \Psi_i$ for each Ψ_i, and $\Sigma \cup \{\Psi_1, \Psi_2, \ldots \Psi_k\} \vdash_T \Phi$, then $\Sigma \vdash_T \Phi$. ■

We now come the main definition and theorem of this section.

> **Definition:** A rule set R is called *inductively local* if there exists some n such that $\mathcal{B}^n[\mathcal{T}_0[R]]$ is self-justifying.
>
> **Theorem:** There exists a procedure which, given any finite set R of Horn clauses, will terminate with a feedback event whenever R is not local, terminate with "success" whenever R is inductively local, and fail to terminate in cases where R is local but not inductively local.

The procedure is derived by lifting the above ground procedure for computing $\mathcal{B}^n[\mathcal{T}[R]]$. Lifting can be formalized as a mechanical operation on arbitrary nondeterministic ground procedures [McAllester and Siskind, 1991]. In the lifted version the infinite set $\mathcal{B}^j[\mathcal{T}_0[R]]$ is represented by a finite set of "template schemas" each of which consists of a template expression $\Sigma, \Gamma \vdash_{T \cup \{\alpha\}} \Phi$ involving variables plus a set of constraints on those variables.

7 LOCALITY IS UNDECIDABLE

We prove that locality is undecidable by reducing the Halting problem. Let T be a specification of a Turing machine. We first show one can mechanically construct a local rule set R with the property that the machine T halts if and only if there exists a term t such that $\vdash_R H(t)$ where H is a monadic predicate symbol. Turing machine computations can be represented by first order terms and the formula $H(t)$ intuitively states that t is a term representing a halting computation of T.

To prove this preliminary result we first construct a superficial rule set S such that T halts if and only if there exists a term t such that $\texttt{INPUT}(t) \vdash_S H(t)$. The mechanical construction of the superficial rule set S from the Turing machine T is fairly straightforward and is not given here. We convert this superficial rule set S to a local rule set R as follows. For each predicate symbol Q of m arguments appearing in S let Q' be a new predicate symbol of $m+1$ arguments. The rule set R will be constructed so that $\vdash_R Q'(t, s_1, \ldots s_m)$ just in case $\texttt{Input}(t) \vdash_S Q(s_1, \ldots s_m)$. We define the rule set R to be the rule set containing the following clauses.

- $\texttt{Input}'(x)$
- All clauses of the form

$$Q_1'(x, t_{1,1}, \ldots t_{1,m_1}) \wedge \cdots \wedge Q_n'(x, t_{n,1}, \ldots t_{n,m_n}) \Rightarrow W'(x, s_1, \ldots s_j)$$

  where the clause $Q_1(t_{1,1}, \ldots t_{1,m_1}) \wedge \cdots \wedge Q_n(t_{n,1}, \ldots t_{n,m_n}) \Rightarrow W(s_1, \ldots s_j)$ is in R.
- The clause $H'(x, x) \Rightarrow H(x)$.

Given that $\vdash_R Q'(t, s_1, \ldots s_m)$ if and only if $\texttt{Input}(t) \vdash_S Q(s_1, \ldots s_m)$ it directly follows that $\texttt{Input}(t) \vdash_S H(t)$ if and only if $\vdash_R H(t)$. So the Turing machine T halts if and only if $\vdash_R H(t)$ for some term t. The proof that the rule set R is local closely follows the proof of the Local Rule Set Representation Theorem proven above.

We have now constructed a local rule set R with the property that T halts if and only if there exists some term t such that $\vdash_R H(t)$. Now let R' be R plus the single clause $H(x) \Rightarrow \texttt{Halts}$ where **Halts** is a new proposition symbol. We claim that R' is local if and only if T does not halt. First note that if T halts then we have $\vdash_{R'} \texttt{Halts}$ but $\not\vdash_{R'} \texttt{Halts}$ so R is not local. Conversely, suppose that T does not halt. In this case we must show that R' is local. Suppose that $\Sigma \vdash_{R'} \Phi$. We must show that $\Sigma \vdash_{R'} \Phi$. Suppose Φ is some formula other than **Halts**. In this case $\Sigma \vdash_{R'} \Phi$ is equivalent to $\Sigma \vdash_R \Phi$. Since R is local we must have $\Sigma \vdash_R \Phi$ and thus $\Sigma \vdash_{R'} \Phi$. Now suppose Φ is the formula **Halts**. If **Halts** is a member of Σ then the result is trivial so we assume that **Halts** is not in Σ. Since $\Sigma \vdash_{R'} \texttt{Halts}$ we must have $\Sigma \vdash_{R'} H(c)$ for some term c. To show $\Sigma \vdash_{R'} \texttt{Halts}$ it now suffices to show that c is mentioned in Σ. By the preceding argument we have $\Sigma \vdash_R H(c)$. Since the rule set R was generated by the construction given above, we have that every inference based on a clause in R is such that that every formula in the inference has the same first argument. This implies that $\Sigma' \vdash_R H(c)$ where Σ' is the subset of formulas in Σ that have c as a first argument. We have assumed that T does not halt, and thus $\not\vdash_R H(c)$. Hence Σ' must not be empty. So Σ must mention c and since $\Sigma \vdash_R H(c)$ we have $\Sigma \vdash_{R'} \texttt{Halts}$.

8 OPEN PROBLEMS

In closing we note some open problems. There are many known examples of rule sets which are not local and yet the corresponding inference relation is polynomial time decidable. In all such cases we have studied there exists a conservative extension of the rule set which is local. We conjecture that for every rule set R such that $\vdash_R$ is polynomial time decidable there exists

a local conservative extension of R. Our other problems are less precise. Can one find a "natural" rule set that is local but not inductively local? A related question is whether there are useful machine recognizable subclasses of the local rule sets other than the classes of bounded local and inductively local rule sets?

Acknowledgement

We would like to thank Franz Baader for his invaluable input and discussions. Support for the work described in this paper was provided in part by Mitsubishi Electric Research Laboratories, Inc. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124. Robert Givan is supported by a Fannie and John Hertz graduate fellowship.

References

[Bry, 1990] Francois Bry. Query evaluation in recursive databases: bottom-up and top-down reconciled. *Data and Knowledge Engineering*, 5:289–312, 1990.

[Givan *et al.*, 1991] Robert Givan, David McAllester, and Sameer Shalaby. Natural language based inference procedures applied to schubert's steamroller. In *AAAI-91*, pages 915–920. Morgan Kaufmann Publishers, July 1991.

[Immerman, 1986] Neal Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

[Jaffar and Lassez, 1987] J. Jaffar and J. L. Lassez. Constraint logic programming. In *Proceedings of POPL-87*, pages 111–119, 1987.

[Knuth, 1975] Donald E. Knuth. Estimating the efficiency of backtrack programs. *Mathematics of Computation*, 29(129):121–136, January 1975.

[Kozen, 1977] Dexter C. Kozen. Complexity of finitely presented algebras. In *Proceedings of the Ninth Annual ACM Symposium on the Theory of Computation*, pages 164–177, 1977.

[Mackworth, 1977] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–181, 1977.

[McAllester and Siskind, 1991] David Allen McAllester and Jeffrey Mark Siskind. Lifting transformations. MIT Artificial Intelligence Laboratory Memo 1343, 1991.

[McAllester, 1989] David A. McAllester. *Ontic: A Knowledge Representation System for Mathematics*. MIT Press, 1989.

[McAllester, 1990] D. McAllester. Automatic recognition of tractability in inference relations. Memo 1215, MIT Artificial Intelligence Laboratory, February 1990. To appear in JACM.

[Naughton and Ramakrishnan, 1991] Jeff Naughton and Raghu Ramakrishnan. Bottom-up evaluation of logic programs. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic*. MIT Press, 1991.

[Papadimitriou, 1985] Christos H. Papadimitriou. A note on the expressive power of prolog. *EATCS Bulletin*, 26:21–23, 1985.

[Pearl and Korf, 1987] Judea Pearl and Richard Korf. Search techniques. *Ann. Rev. Comput. Sci.*, 2:451–467, 1987.

[Shostak, 1978] R. Shostak. An algorithm for reasoning about equality. *Comm. ACM.*, 21(2):583–585, July 1978.

[Ullman, 1988] J. Ullman. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, 1988.

[Ullman, 1989] J. Ullman. Bottom-up beats top-down for datalog. In *Proceedings of the Eigth ACM SIGACT-SIGMOD-SIGART Symposium on the Principles of Database Systems*, pages 140–149, March 1989.

[van Hentenryck, 1989] Pascal van Hentenryck. *Constraint Satisfaction in Logic Programming*. MIT Press, 1989.

[Vardi, 1982] M. Vardi. Complexity of relational query languages. In *14th Symposium on Theory of Computation*, pages 137–146, 1982.

An Order-Sorted Logic with Sort Literals and Disjointness Constraints

Toni Bollinger, Udo Pletat
IBM Deutschland GmbH
AE Software Architekturen und Technologien
Postfach 800880
W-7000 Stuttgart 80, Germany
email: bollinger@vnet.ibm.com, pletat@ds0lilog.bitnet

Abstract

We present a resolution calculus for an order sorted logic where sorts can be used as unary predicates and where constraints about the disjointness of two sorts can be stated in the sort signature besides constraints specifying the subsort-relationship. Besides providing a detailed completeness proof, we outline a refinement of the calculus that limits certain negative effects on the search space compared to standard order sorted logics.

1 Introduction

Hybrid logics combining a logic describing sorts with first order predicate logic have been investigated extensively in the recent past, e.g., see [Oberschelp, 1962], [Walther, 1987], [Schmidt-Schauß, 1989], [Frisch, 1991], [Cohn, 1987], [Cohn, 1992], [Beierle *et al.*, 1992], or [Weidenbach and Ohlbach, 1990]. In this context one may distinguish essentially two frameworks:

- the *substitutional framework* [Frisch, 1991], where the information concerning sorts can be represented by an axiomatization employing only sorts;
- the *non-substitutional framework*, where no restrictions on the axiomatization about sorts apply.

Logics of the non-substitutional framework are characterized by the fact that they allow sort literals, where a sort is used as a unary predicate, to occur arbitrarily in the formulas. In the substitutional framework, the sort axiomatization is strictly separated from the rest of a knowledge base, where sorts are associated with variables only and are taken into account during a resolution proof only when unifying literals.

Using sort literals, a logic gains expressive power as this enables us to impose conditions on the sort membership of a term that can't be expressed with sorts only. As a drawback, however, one looses efficiency, as the deductions become more complex. It is beyond the scope of this paper to discuss the pro and cons of sort literals in detail. This has already been done elsewhere, e.g., in [Cohn, 1989].

In this paper we present the Σ_{DSL}-logic[1], a logic belonging to the non-substitutional framework. The Σ_{DSL}-logic is an extension of the logic presented in [Beierle *et al.*, 1992], as we allow also disjointness constraints on sorts, besides constraints specifying the subsort-relationship between two sorts. In the classical order-sorted framework, disjointness constraints are useless, as the sort of a term is static. However, provided that the sort of a term can be defined by axiomatic information, they allow us to specify to which sorts a certain term never belongs.

The inference calculus for the Σ_{DSL}-logic is an improved version of the calculus provided in [Beierle *et al.*, 1992] by taking into account the disjointness constraints for sorts and by indroducing the notion of a maximally general refuting substitution which improves the implementability of the calculus.

The main contributions of this paper are twofold:

1. We provide a new completeness proof that corrects the faulty lifting arguments in [Beierle *et al.*, 1992] and [Weidenbach and Ohlbach, 1990], c.f. Section 3.3.4.
2. We outline a refinement of the calculus such that its computational behaviour can get quite close to that of order sorted logic if the initial number of (positive) sort literals is low, c.f. Section 3.4.

The paper is organized as follows: first we present the syntax and semantics of the Σ_{DSL}-logic and after introducing some basic notions we describe the inference calculus. Then we show its soundness and completeness. We conclude with a discussion of related work.

[1] "D" stands for disjointness constraints, and "SL" for sort literals.

2 Syntax and Semantics of the Σ_{DSL}-Logic

A *signature* within our Σ_{DSL}-logic is a quadrupel $\Sigma = (\mathcal{S}, \mathcal{SC}, \mathcal{P}, \mathcal{F})$, where

- $\mathcal{S}$ is a set of sort symbols;
- $\mathcal{SC}$ is a set of sort constraints for $\mathcal{S}$ of the form $S_1 < S_2$ (S_1 is a subsort of S_2) or $S_1 \perp S_2$ (S_1 and S_2 are disjoint), where $S_1, S_2 \in \mathcal{S}$;
- $\mathcal{P} = \bigcup_i \mathcal{P}^i$, where $\mathcal{P}^i$ is the set of predicate symbols of arity i, such that $\mathcal{S} \subseteq \mathcal{P}^1$; i.e, sorts can be used as unary predicate symbols too;
- $\mathcal{F} = \bigcup_i \mathcal{F}^i$, where $\mathcal{F}^i$ is the set of function symbols of arity i, and where every $f \in \mathcal{F}^i$ is associated with a (range) sort $S \in \mathcal{S}$ denoted by $f : S$.

Besides the sort symbols in $\mathcal{S}$ we may use $\top$, the top sort, standing for everything, and $\perp$, the bottom sort that has no elements. We denote by $\leq$ the transitive and reflexive closure of $<$. Further, let $S_1 \underline{\perp} S_2$, if there is a sort constraint $S_3 \perp S_4 \in \mathcal{SC}$ (or $S_4 \perp S_3 \in \mathcal{SC}$) with $S_1 \leq S_3$ and $S_2 \leq S_4$.

Terms and formulas are built in the usual way over a signature by using this vocabulary together with sorted variables as well as the logical connectives and quantifiers. If a literal has a sort as its predicate symbol we call it a *sort literal*. They can occur arbitrarily in any formula.

The *syntactic sort* of a term is determined by the sort of its outermost function symbol; i.e., for $t = f(t_1, ..., t_n)$ with $f : S \in \mathcal{F}^n$ we have $sort(t) = S$, for a variable $x : S'$ we let $sort(x) = S'$.

An *interpretation* I of a signature Σ consists of a universe U and associates with every sort $S \in \mathcal{S}$ a non-empty set $U_S \subseteq U$. The top sort $\top$ is interpreted by $U_\top = U$ and the bottom sort $\perp$ by $U_\perp = \emptyset$. Further, I assigns a total function $f_I : U^n \to U_S$ to every n-ary function symbol $f : S$ in Σ, and a relation $P_I \subseteq U^n$ to every predicate symbol $P \in \mathcal{P}^n$ in Σ. Sorts being used as predicates are interpreted such that $a \in U_S$ iff $(a) \in S_I$ holds.

I is a *model* of Σ (called Σ-model in the sequel), if for every sort constraint $S_1 < S_2 \in \mathcal{SC}$ we have $U_{S_1} \subseteq U_{S_2}$ and for every sort constraint $S_3 \perp S_4 \in \mathcal{SC}$ $U_{S_3} \cap U_{S_2} = \emptyset$ holds. A signature Σ is *consistent*, if Σ has a model. An inconsistency arises, if for two sorts S and S' both $S \leq S'$ and $S \underline{\perp} S'$ hold, such that we have $U_S = \emptyset$.

Interpretations are extended in the usual way to terms and formulas. If a formula F holds under all Σ-models we denote it by $\models_\Sigma F$. Further, if all Σ-models satifying a formula F_1 satisfy also a formula F_2 this is denoted by $F_1 \models_\Sigma F_2$.

One can show that all Σ-models satisfy the axioms representing the sort contraints; i.e, we have
$\models_\Sigma \forall x : S_1 \;\; S_2(x)$ for $S_1 < S_2 \in \mathcal{SC}$ and
$\models_\Sigma \forall x : S_3 \;\; \neg S_4(x)$ for $S_3 \perp S_4 \in \mathcal{SC}$.

If we assume Σ to be consistent, i.e., if no sort has to interpreted by the empty sort, we can transform any formula F to its clausal normal form without changing its satisfiability. In the following, we consider only such sets of clauses $\mathcal{C}$ together with its associated signature Σ. .

3 A Calculus for the Σ_{DSL}-Logic

3.1 Basic Notions

Let us first introduce some basic notions. We consider an *expression* to be either a literal or a term. For an expression e, $var(e)$ denotes the set of variables occuring in e.

Given a substitution $\sigma = \{x_1 \leftarrow t_1, ..., x_n \leftarrow t_n\}$, $x_i \neq t_i$, $i = 1, ..., n$, we call $domain(\sigma) = \{x_1, ..., x_n\}$ the *domain* of σ, $range(\sigma) = \{t_1, ..., t_n\}$ the *range* of σ, $varrange(\sigma) = \{y \in range(\sigma) \mid y \text{ is a variable}\}$ is the set of variables being an element of the range of σ. Note this is different from $var(range(\sigma))$ which includes also the variables occuring in other terms of $range(\sigma)$. Further, ϵ denotes the empty substitution. The composition $\sigma\varrho$ of two substitutions σ and ϱ is defined in the usual way (cf. [Loveland, 1978]), such that $\sigma\varrho(e) = \varrho(\sigma(e))$ for an expression e. $\alpha = \{x_1 \leftarrow y_1, ..., x_n \leftarrow y_n\}$ is a *variable renaming* iff each y_i is a variable and $y_i \neq y_j$ for $i \neq j$. The inverse $\{y_1 \leftarrow x_1, ..., y_n \leftarrow x_n\}$ of a variable renaming is denoted by α^{-1}. Note that $\alpha\alpha^{-1} = \alpha^{-1}$, even though we have for an expression e with $var(e) \cap range(\alpha) = \emptyset$ $(\alpha\alpha^{-1})(e) = e$. σ' is a *variant* of σ, if $\sigma' = \sigma\alpha$ for a variable renaming α with $domain(\alpha) \subseteq varrange(\sigma)$.

Disjointness contraints have to be taken into account so that substitutions denote valid replacements. Obviously, the sort of a variable and its replacing term should not be disjoint. Additionally, if two variables are replaced by the same term, the sorts of the replaced variables should not be disjoint too.

Definition 1 (DISJ-Substitution) [2]
A substitution σ is a DISJ-substitution, iff we have for all $t \in range(\sigma)$ and for all x_1, $x_2 \in domain(\sigma)$ with $x_1 \leftarrow t \in \sigma$, $x_2 \leftarrow t \in \sigma$:
$sort(x_1) \not\perp sort(t)$, $sort(x_2) \not\perp sort(t)$, as well as $sort(x_1) \not\perp sort(x_2)$.

DISJ-substitutions σ are not necessarily syntactically well-typed, i.e., we may have $sort(t) \not\leq sort(x)$ for $x \leftarrow t \in \sigma$. In such a case, the semantic well-typedness

[2] DISJ-substitutions are sort consistent in Cohn's terminology, cf. [Cohn, 1992].

of σ depends also on the condition that $S_x(t)$ is a valid formula for $sort(x) = S_x$. Therefore, when instantiating a clause C by a DISJ-substitution being not well-typed, the negation of these conditions have to be added to C.

Definition 2 (Instantiation)
Given a DISJ-substitution σ
$[\sigma]C = \sigma(C) \vee \neg SL(\sigma)$ *is an instance of* C,
where $SL(\sigma)$ *is a conjunction of sort literals satisfying the following condition:*
$S_x(t) \in SL(\sigma)$ *iff*
$S_x = sort(x)$, $x \leftarrow t \in \sigma$ *and* $sort(t) \not\leq sort(x)$.

For defining the rules of our calculus we need the notions of *complementarity of literals* and *refuting substitution.*

Definition 3 (Complementarity of Literals)
Two literals L_1 *and* L_2 *are complementary, iff one of the following conditions holds:*

- $L_1 = \neg L_2$,
- $L_1 = \neg S_1(t)$, $L_2 = S_2(t)$, *and* $S_2 \leq S_1$,
- $L_1 = S_1(t)$, $L_2 = S_2(t)$, *and* $S_1 \perp S_2$.

A literal L_3 *is self-complementary, iff*

- $L_3 = \neg S(t)$ *and* $sort(t) \leq S$, *or*
- $L_3 = S(t)$ *and* $sort(t) \perp S$.

One can show for complementary and self-complementary literals: $\models_\Sigma \neg(L_1 \wedge L_2)$ resp. $\models_\Sigma \neg L_3$.

During a refutation proof literals are made complementary by applying substitutions to them. We call such substitutions *refuting substitutions.*

Definition 4 (Refuting Substitution)
A DISJ-substitution σ *is a* refuting-substitution *of*

- *two literals* L_1 *and* L_2 *iff*
  $\sigma(L_1)$ *and* $\sigma(L_2)$ *are complementary,*
- *a single literal* L_3, *iff*
  $\sigma(L_3)$ *is self-complementary.*

A refuting substitution of two literals L_1 and L_2 is either a unifier of L_1 and $\neg L_2$, or, if L_1 and L_2 are sort literals, it is a unifier of their arguments. If we disregard the sorts, we know that there is a most general unsorted unifier. We refer to it by *most general unsorted refuting substitution. Maximally general refuting substitutions*[3] which will be defined below are variants of them. The sort of a replacing variable is essentially a maximal subsort of the sorts of a subset of the replaced variables. For the completeness of the calculus it is necessary to consider all such subsets (cf. Example 6 in Section 4).

Definition 5 (Max. General Ref. Substitution)
A refuting substitution Θ *of two literals* L_1 *and* L_2 *is a* maximally general *refuting substitution iff*

- *considering* Θ *as an unsorted substitution it is a variant of a most general unsorted refuting substitution;*
- *for any variable* $y \in range(\Theta)$, *let*
  $X = \{x \in domain(\Theta) \mid \Theta(x) = y\}$
  be the set of variables y *replaces;*
  then there has to be a set $X' \subseteq X$ *such that*
  - $sort(y)$ *is a maximal common subsort of* $\{sort(x') \mid x' \in X'\}$,
  - $sort(y) \neq \perp$,
  - *there is no* $\overline{x} \in X$ *such that*
    $X' \subset \{x' \in X \mid sort(\overline{x}) \leq sort(x')\}$[4].

Θ *is a maximally general refuting substitution of a sort literal* L_3 *iff*

- *if* L_3 *is self-complementary, then* $\Theta = \epsilon$;
- *if* L_3 *is not self-complementary and the argument of* L_3 *is a variable* $x : S_x$,
  then $\Theta = \{x \leftarrow y : S_y\}$ *and* S_y *is a maximal common subsort of* $\{S_x, S_R\}$ *or* $\{S_R\}$, *where* S_R *is a maximally general sort such that* $\{x \leftarrow y : S_R\}(L_3)$ *is self-complementary.*

We have to consider variants of most general unsorted refuting substitutions, as the specialization of the sorts of variables obliges us to introduce new variables.

Concerning maximally general refuting substitutions for single literals one might miss the case where the argument of a sort literal is not a variable. However, this case is covered by the first alternative, as the sort of a non-variable term being determined by the sort of its outermost function symbol cannot be specialized by instantiating it. Concerning the second alternative, we have $S_R = S$ for $L_3 = \neg S(x)$ and $S_R = S_1$ for $L_3 = S(x)$ with $S \perp S_1$ such that $S \not\perp S_2$ for all $S_1 \leq S_2$.

The following example illustrates our definition:

[3] In a strict sense, if we use the notion of "maximally general" we should define a partial order on the set of refuting substitutions. Lemma 1 gives us a hint for defining such a partial order, and it can be shown that, if we define this relation appropriately, maximally general refuting substitutions satisfy the conditions of Definition 5.

[4] This condition restricts the potential number of maximally general refuting substitutions. It assures also that these refuting substitutions are maximal according to the partial order mentionned in the preceeding footnote.

Example 1 *Given* $\Sigma = \langle \mathcal{S}, \mathcal{SC}, \mathcal{P}, \mathcal{F} \rangle$ *with*
$\mathcal{S} = \{S_1, S_2, S_3, S_4\}$,
$\mathcal{SC} = \{S_3 < S_1, S_3 < S_2, S_4 < S_1, S_4 < S_2, S_3 \perp S_4\}$,
$\mathcal{P} = \mathcal{P}^2$ *with* $\mathcal{P}^2 = \{Q\}$, *and* $\mathcal{F} = \emptyset$.

For $Q(x_1 : S_1, x_3 : S_3)$ *and* $\neg Q(x_2 : S_2, x_2 : S_2)$ $\Theta' = \{x_1 \leftarrow x_2, x_3 \leftarrow x_2\}$ *is a most general unsorted refuting substitution. The following variants of* Θ' *are candidates for a maximally general refuting substitution:*
$\Theta_1 = \{x_1 \leftarrow y_1 : S_1, x_2 \leftarrow y_1, x_3 \leftarrow y_1\}$,
$\Theta_2 = \{x_2 \leftarrow y_2 : S_2, x_2 \leftarrow y_2, x_3 \leftarrow y_2\}$,
$\Theta_3 = \{x_1 \leftarrow y_3 : S_3, x_2 \leftarrow y_3, x_3 \leftarrow y_3\}$,
$\Theta_4 = \{x_1 \leftarrow y_4 : S_4, x_2 \leftarrow y_4, x_3 \leftarrow y_4\}$.

With $X = \{x_1, x_2, x_3\}$ *we have for* $\{x_1\} \subseteq X$ *trivially that* S_1 *is a maximal common non-empty subsort of* $sort(x_1) = S_1$. *However, for* $x_3 : S_3$ *we have* $\{x_1\} \subset \{x \in X \mid S_3 \leq sort(x)\} = X$, *such that* Θ_1 *is not a maximally general refuting substitution. For the same reason this isn't the case neither for* Θ_2 *nor for* Θ_4. *The only maximally general refuting substitution is therefore* Θ_3.

The literal $\neg S_1(x_2 : S_2)$ *has three maximally general refuting substitutions* ($S_R = S_1$):
$\Theta_5 = \{x_2 \leftarrow y_5 : S_3\}$,
$\Theta_6 = \{x_2 \leftarrow y_6 : S_4\}$,
$\Theta_7 = \{x_2 \leftarrow y_7 : S_1\}$;
and $S_3(x_1 : S_1)$ *has only one* ($S_R = S_4$):
$\Theta_8 = \{x_1 \leftarrow y_8 : S_4\}$.

We need the following lemma for proving the lifting lemma in Section 3.3.4:

Lemma 1 (Decomposition Lemma)
If σ *is refuting substitution of two literals* L_1 *and* L_2 *or a single literal* L_3 *then there is a maximally general refuting substitution* Θ *of* L_1 *and* L_2 *resp.* L_3 *and a substitution* ϱ *such that*

- $\Theta\varrho \supseteq \sigma$,
  i.e., for every $x \in domain(\sigma)$ *we have* $\sigma(x) = \Theta\varrho(x)$, *and*
- $\neg\varrho SL(\Theta) \cup \neg SL(\varrho) \subseteq \neg SL(\sigma)$.[5]

Proof: *We consider first the case where* Θ *is a maximally general refuting substitution of* L_1 *and* L_2. *If we consider* Θ *as an unsorted substitution there is a most general unsorted refuting substitution* Θ' *and an unsorted substitution* ϱ' *such that* $\Theta'\varrho' = \sigma$. *Let* α *be a variable renaming with* $domain(\alpha) = varrange(\Theta')$ *and* $range(\alpha) \cap (var(L_1) \cup var(L_2)) = \emptyset$, *i.e.,* α *replaces the variables in* $range(\Theta')$ *by new variables not occurring in* L_1 *and* L_2, *and thus not in* Θ'. *Hence,* $\Theta'\alpha^{-1} = \Theta' \cup \alpha^{-1} \supseteq \Theta'$ *due to* $domain(\alpha^{-1}) = range(\alpha)$. *Thus* $\Theta'\alpha^{-1}\varrho' \supseteq \Theta'\varrho'$, *and due to* $\alpha\alpha^{-1} = \alpha^{-1}$ *we have* $\Theta'\alpha\alpha^{-1}\varrho' \supseteq \Theta'\varrho'$. *Hence with* $\Theta = \Theta'\alpha$ *and* $\varrho = \alpha^{-1}\varrho'$ *we obtain* $\Theta\varrho \supseteq \sigma$.

We still have to determine the sorts of the replacing variables such that on the one side Θ *and* ϱ *are DISJ-substitutions and* Θ *satisfies the conditions for a maximally general refuting substitution and such that on the other side the statement of the lemma holds. In the following we denote where appropriate by* S_t *the sort of a term* t.

We first show that $\neg\varrho SL(\Theta) \subseteq \neg SL(\sigma)$ *holds. Given* $x \leftarrow y \in \Theta$, *we have to consider the following cases:*

1. $x \notin domain(\sigma)$,
   i.e., we have $\sigma(x) = x$; *y has to be a variable and* $y \leftarrow x \in \varrho$. *With* $sort(x) = sort(y)$ *neither* $\neg S_x(y) \in \neg SL(\Theta)$ *nor* $\neg S_y(x) \in \neg SL(\varrho)$.

2. $x \in domain(\sigma)$, *y is not a variable.*
   Thus $sort(\varrho(y)) = S_y$ *and* $x \leftarrow \varrho(y) \in \sigma$.

   Consequently $S_x \not\perp S_y$, *as* σ *is a DISJ-substitution. For* x' *with* $x' \leftarrow y \in \Theta$, *we have* $x' \neq \varrho(y)$ *due to* $x' \leftarrow \varrho(y) \in \sigma$. *Hence* $sort(x') \not\perp S_x$, *and* Θ *fulfils the conditions for a DISJ-substitution.*

   Concerning the sort literals, $\neg S_x(y) \in \neg SL(\Theta)$ *entails* $S_y \not\leq S_x$. *Hence* $sort(\varrho(y)) \not\leq S_x$ *holds too, such that* $\neg S_x(\varrho(y)) \in \neg SL(\sigma)$.

3. $x \in domain(\sigma)$, *y is a variable,*
   i.e., there is a term $z \in range(\sigma)$ *such that* $x \leftarrow z \in \sigma$ *and* $y \leftarrow z \in \varrho$.

   Let $X = \{x' \in domain(\Theta) \mid x' \leftarrow y \in \Theta\}$ *be the set of variables y replaces in* Θ *and let* $X_z = \{x' \in X \mid S_z \leq sort(x')\}$.

   If $X_z \neq \emptyset$, *we set* S_y *to a maximal common subsort of the sorts of the variables in* X_z, *unless there is a variable* $\overline{x} \in X$ *such that* $X_z \subseteq \{x' \in X \mid sort(\overline{x}) \leq sort(x')\}$ *and such that we have* $sort(x') < sort(\overline{x})$ *for no variable* $x' \in X$. *In this case* $S_y = sort(\overline{x})$.
   If $X_z = \emptyset$, $S_y = sort(\overline{x})$ *with* $\overline{x} \in X$ *and* $sort(x') \not\leq sort(\overline{x})$ *for all* $x' \in X$.

   Obviously, S_y *fulfils the conditions of Definition 4. The conditions for DISJ-substitutions are also satisfied. First*
   (*) $X \subseteq \{x' \in domain(\sigma) \mid x' \leftarrow z \in \sigma\} \cup \{z\}$ *holds, as* $x' \in X$ *and* $x' \notin domain(\sigma)$ *leads to* $\sigma(x') = x' = z$ *due to* $x' \leftarrow z \in \Theta\varrho$ *and* $\Theta\varrho \supseteq \sigma$.
   As σ *is a DISJ-substitution, we have due to (*) for* $x_1, x_2 \in X$ $sort(x_1) \not\perp sort(x_2)$ *as well as* $S_z \not\perp sort(x_1)$. *Further, due to our construction of* S_y, *we have* $S_z \leq S_y$ *or* $S_y = sort(\overline{x})$, *for* $\overline{x} \in X$, *which leads in both cases to* $sort(x_1) \not\perp S_y$.

   For checking the sort literals let us assume $\neg S_x(y) \in \neg SL(\Theta)$ *and* $\neg S_x(z) \notin \neg SL(\sigma)$ *corre-*

[5] The inclusion relation in the other direction has been proven in [Beierle *et al.*, 1992] already.

sponding to $S_z \leq S_x$. This isn't possible, however, as $S_z \leq S_x$ entails $x \in X_z$, such that according to our construction $S_y \leq S_x$ would also hold.

Note, for determining the sort of y we have considered all the variables y replaces. That's why the arguments above are valid also for the other variables $x^* \in X$ with $x^* \neq x$.

We still have to check the sort literals of ϱ, i.e., we have to show $\neg SL(\varrho) \subseteq \neg SL(\sigma)$.
$\Theta = \Theta'\alpha$ and $varrange(\Theta') = domain(\alpha)$ entail $varrange(\Theta) = range(\alpha)$; $\Theta'\varrho' = \sigma$ yields with $\varrho' \subseteq \sigma$ $domain(\varrho') \subseteq domain(\sigma)$. Further, $\alpha^{-1}\varrho' = \varrho$ entails $domain(\varrho) \subseteq range(\alpha) \cup domain(\varrho')$, such that $domain(\varrho) \subseteq varrange(\Theta) \cup domain(\sigma)$.

Now given $y \leftarrow z \in \varrho$ we have for $y \in varrange(\Theta)$ according to the construction of the sort of y above (cases 1 and 3): $S_z \leq S_y$ or $S_y = sort(\overline{x})$ with $\overline{x} \leftarrow z \in \sigma$. This covers also the conditions on S_y, if $y \in domain(\sigma)$ as this entails $y \leftarrow z \in \sigma$.

Thus obviously $S_x \not\perp S_y$. For $y' \leftarrow z \in \varrho$ $S_y \not\perp S_{y'}$ holds too. This can be checked by a simple case analysis.

$\neg S_y(z) \in \neg SL(\varrho)$, i.e., $S_z \not\leq S_y$, is only possible if $S_y = sort(\overline{x})$. As $\overline{x} \leftarrow z \in \sigma$, $\neg S_y(z) \in \neg SL(\sigma)$ holds too.

Finally, let us consider the case where σ is a refuting substitution of L_3:

1. *If L_3 is self-complementary, then $\Theta = \epsilon$ and the lemma trivially holds with $\varrho = \sigma$.*

2. *If L_3 is not self-complementary, then the argument of L_3 has to be a variable;*
   i.e., $L_3 = S(x : S_x)$ or $L_3 = \neg S(x : S_x)$,
   $\sigma = \{x \leftarrow t\}$, $\Theta = \{x \leftarrow y\}$ and $\varrho = \{y \leftarrow t\}$.
   As $\sigma(L_3)$ is self-complemtary we have $sort(t) \leq S_R$.

   (a) $sort(t) \leq S_x$:
   S_y is a maximal common subsort of S_x and S_R.
   Hence $S_y \leq S_x$ and $sort(t) \leq S_y$, as sort(t) is a subsort of S_R and S_x. Thus Θ as well as ϱ are DISJ-substitutions and we have $\neg SL(\Theta) = \neg SL(\varrho) = \neg SL(\sigma) = \Box$.

   (b) $sort(t) \not\leq S_x$:
   $S_y = S_R$.
   This together with $sort(t) \leq S_R$ and $sort(t) \not\perp S_x$ entail $S_y \not\perp S_x$ and $sort(t) \not\perp S_y$.
   $S_x \not\leq S_R$ as L_3 is not self-complementary.
   Hence $\neg SL(\Theta) = \neg S_x(y)$, $\neg SL(\varrho) = \Box$ and $\neg SL(\sigma) = \neg S_x(t)$.

□

Example 2 *Given $\Sigma = \langle \mathcal{S}, \mathcal{SC}, \mathcal{P}, \mathcal{F} \rangle$ with*
$\mathcal{S} = \{S_1, S_2, S_3, S_4\}$,
$\mathcal{SC} = \{S_3 < S_1, S_3 < S_2, S_4 < S_1, S_4 < S_2\}$,
$\mathcal{P} = \mathcal{P}^1 \cup \mathcal{P}^2$ with $\mathcal{P}^1 = \{P\}$, $\mathcal{P}^2 = \{Q\}$, and
$\mathcal{F} = \mathcal{F}^0 = \{a_2 : S_2, a_3 : S_3, a_4 : S_4\}$.

For $P(x : S_1)$ and $\neg P(y : S_2)$
$\sigma = \{x \leftarrow a_2, y \leftarrow a_2\}$ is a refuting substitution with $\neg SL(\sigma) = \neg S_1(a_2)$,
and these are the maximally general refuting substitutions:
$\Theta_1 = \{x \leftarrow z_1 : S_1, y \leftarrow z_1\}$ with $\neg SL(\Theta_1) = \neg S_2(z_1)$,
$\Theta_2 = \{x \leftarrow z_2 : S_2, y \leftarrow z_2\}$ with $\neg SL(\Theta_2) = \neg S_1(z_2)$,
$\Theta_3 = \{x \leftarrow z_3 : S_3, y \leftarrow z_3\}$ with $\neg SL(\Theta_3) = \Box$,
$\Theta_4 = \{x \leftarrow z_4 : S_4, y \leftarrow z_4\}$ with $\neg SL(\Theta_4) = \Box$.
Lemma 1 holds only for Θ_2 with $\varrho_2 = \{z_2 \leftarrow a_2\}$.

For $Q(x_1 : S_1, x_3 : S_3)$ and $\neg Q(x_2 : S_2, x_2 : S_2)$
$\sigma_3 = \{x_1 \leftarrow a_3, x_2 \leftarrow a_3, x_3 \leftarrow a_3\}$ as well as
$\sigma_4 = \{x_1 \leftarrow a_4, x_2 \leftarrow a_4, x_3 \leftarrow a_4\}$
are refuting substitutions, and $\neg SL(\sigma_3) = \Box$ resp. $\neg SL(\sigma_4) = \neg S_3(a_4)$.
$\Theta_5 = \{x_1 \leftarrow z_5 : S_3, x_2 \leftarrow z_5, x_3 \leftarrow z_5\}$
is the only maximally general refuting substitution (cf. Example 1) with $\neg SL(\Theta_5) = \Box$, and the lemma holds both for σ_3 and σ_4 (for which $\varrho_4 = \{z_5 \leftarrow a_4\}$ with $\neg SL(\varrho_4) = \neg S_3(a_4)$).

3.2 The Calculus Rules

Now we are able to define the calculus rules for our Σ_{DSL}-logic.

Definition 6 (Σ_{DSL}- Calculus Rules)
Let C_1 and C_2 be clauses, and let L_1 and L_2 resp. L_3 be literals that can be made complementary by a maximally general refuting substitution Θ. Then the Σ_{DSL}-calculus rules are defined as follows:

- **Σ_{DSL}-resolution**:

$$\frac{L_1 \vee C_1, L_2 \vee C_2}{\Theta(C_1 \vee C_2) \vee \neg SL(\Theta).}$$

- **Σ_{DSL}-factorization**:

$$\frac{\neg L_1 \vee L_2 \vee C}{\Theta(L_2 \vee C) \vee \neg SL(\Theta)}$$

- **Σ_{DSL}-elimination**:

$$\frac{L_3 \vee C_1}{\Theta(C_1) \vee \neg SL(\Theta)}$$

Applying the elimination rule on $\neg S_1(x : S_2)$ yields $\neg S_2(y : S_1)$, if neither $S_1 \leq S_2$ nor $S_2 \leq S_1$ holds. We call this instance of the elimination rule the *reformulation case.*

3.3 Soundness and Completeness

Given an input clause set $\mathcal{C}$ we define

$Inst(\mathcal{C}) = \{[\sigma]C \mid C \in \mathcal{C} \text{ and } \sigma \text{ ground}\}$,

where $[\sigma]C = \sigma(C) \vee \neg SL(\sigma)$; i.e., $Inst(\mathcal{C})$ contains all (semantical well-typedness assuring) ground instances of clauses in $\mathcal{C}$. Further let $I_{DSL}(\mathcal{C})$ be the smallest set containing $\mathcal{C}$ and being closed under applications of the Σ_{DSL}-resolution, Σ_{DSL}-factorization and Σ_{DSL}-elimination rule.

For a set of clauses $\mathcal{C}$ whose signature Σ is known to be consistent, the soundness and completeness of the calculus follows from the cycle of implications below:

$$\begin{array}{ccc} \mathcal{C} \text{ satisfiable} & \overset{(1)}{\Rightarrow} & \square \notin I_{DSL}(\mathcal{C}) \\ (2) \Uparrow & & \Downarrow (4) \\ Inst(\mathcal{C}) \text{ satisfiable} & \overset{(3)}{\Leftarrow} & \square \notin I_{DSL}(Inst(\mathcal{C})) \end{array}$$

(1) addresses the soundness of the calculus, (2) states that the satisfiability of the ground instances can be lifted to the predicate logic level, (3) expresses the completeness of the ground calculus, and (4) means that deductions of the empty clause on the ground level can be lifted to the predicate logic level. The diagram shows that all four statements are equivalent and we have in particular

$$\mathcal{C} \text{ is consistent iff } \square \notin I_{DSL}(\mathcal{C})$$

as the soundness and completeness statement.

3.3.1 Soundness

First we show the soundness of our inference rules for the predicate logic level.

Lemma 2 (Soundness of the Calculus Rules)
Let C_1 and C_2 be clauses, L_1 and L_2, resp. L_3 be literals that can be made complementary by a maximally general refuting substitution Θ. Then we have:

1. *Soundness of the Σ_{DSL}-resolution rule:*
   $(L_1 \vee C_1) \wedge (L_2 \vee C_2) \models_\Sigma \Theta(C_1 \vee C_2) \vee \neg SL(\Theta)$
2. *Soundness of the Σ_{DSL}-factorization rule:*
   $\neg L_1 \vee L_2 \vee C_1 \models_\Sigma \Theta(L_2 \vee C_1) \vee \neg SL(\Theta)$
3. *Soundness of the Σ_{DSL}-elimination rule:*
   $L_3 \vee C_1 \models_\Sigma \Theta(C_1) \vee \neg SL(\Theta)$

Proof: *The proof relies on the fact that $C \wedge SL(\sigma) \models_\Sigma \sigma(C)$ leads to $C \models_\Sigma \sigma(C) \vee \neg SL(\sigma)$ and that we have for two complementary literals L_1, L_2, resp. one self-complementary sort literal L_3: $\models_\Sigma \neg\Theta(L_1 \wedge L_2)$ resp. $\models_\Sigma \neg\Theta(L_3)$.*

The soundness theorem follows from Lemma 2 by induction on the definition of $I_{DSL}(\mathcal{C})$.

Theorem 1 (Soundness of the Σ_{DSL}-Calculus)
If a set of clauses $\mathcal{C}$ is consistent then $\square \notin I_{DSL}(\mathcal{C})$.

3.3.2 Lifting Satisfiability

Theorem 2 (Lifting Satisfiability)
For any set of clauses $\mathcal{C}$ the satisfiability of $Inst(\mathcal{C})$ implies that $\mathcal{C}$ is satisfiable.
Proof: *Let M be a term generated model satisfying $Inst(\mathcal{C})$. The proof consists of constructing from M a Σ-model satisfying $\mathcal{C}$.*

3.3.3 Completeness of the Σ_{DSL}- Ground Calculus

For the ground version of our calculus we use a restricted version of the factorization rule, namely that we allow it to eliminate *identical* literals only. This facilitates slightly our argumentation for the lifting proof.

Theorem 3 (Ground Completeness)
Let $\mathcal{C}$ be a set of clauses. If $\square \notin I_{DSL}(Inst(\mathcal{C}))$ then $Inst(\mathcal{C})$ is consistent, i.e., there is a Σ-model M with $M \models_\Sigma Inst(\mathcal{C})$.
Proof: *We prove the theorem by constructing a Herbrand model M satisfying Σ and $Inst(\mathcal{C})$. The general idea behind the model construction is to define a model that satisfies a minimal number of positive sort literals on the one side and on the other side a maximal number of positive non-sort literals. The reason why we construct a minimal model of Σ is that we want to avoid that a disjointness constraint forces us to withdraw the positive interpretation of a sort literal*[6].

For a clause C let $np(C)$ be defined as the sum of the number of negative sort literals and the number of positive non-sort literals in C, i.e., in particular we have $np(C) = 0$ if C consists only of positive sort literals and negative non-sort literals.

Further, let A_1, A_2, A_3, ... be an enumeration of the atomic formulas in $Inst(\mathcal{C})$ where $S_1(t)$ appears after $S_2(t)$, if $S_2 < S_1$. Beginning with $M_0 = \emptyset$, we determine inductively the model M_n from M_{n-1} by defining the truthvalue of A_n.

- *If A_n is a sort literal $S(t)$, $S(t)$ is interpreted* <u>false</u> *unless:*
  - *$sort(t) \leq S$,*
  - *there is a clause $C = C' \vee S'(t) \in I_{DSL}(Inst(\mathcal{C}))$ with $np(C') = 0$, $S' \leq S$, and*

[6] In the ground completeness proof in [Beierle *et al.*, 1992] a maximal model of Σ is constructed. This is sufficient as their sort language allows no disjointness constraints.

where we have for all the atomic formulas A_j of the literals in C': $j < n$; i.e., they have been assigned a truthvalue already, and $M_{n-1} \models \neg C'$.

- *If A_n is a not a sort literal, A_n is interpreted true, unless there is a clause $C = C' \vee \neg A_n \in I_{DSL}(Inst(\mathcal{C}))$ with $np(C') = 0$, and where we have for all the atomic formulas A_j of the literals in C': $j < n$ and $M_{n-1} \models \neg C'$.*

With

$$M = \lim_{n \to \infty} M_n$$

we have to show that M is a Σ-model satisfying $Inst(\mathcal{C})$, or more precisely:

1. *$M \models \Sigma$,*
2. *$M \models Inst(\mathcal{C})$.*

First, we prove $M \models Inst(\mathcal{C})$ by showing $M \models C$ for all $C \in I_{DSL}(Inst(\mathcal{C}))$. This is done by induction on $np(C)$.

1. *Let for $C \in I_{DSL}(Inst(\mathcal{C}))$ $np(C) = 0$, and let us assume $M \models \neg C$. Due to $\square \notin I_{DSL}(Inst(\mathcal{C}))$, $C \neq \square$. Let L be the literal in C whose atom A occurs as the last one in the enumeration, i.e., $C = L \vee C'$ and $L = A$, if L is a sort literal and $L = \neg A$ if L is a non-sort literal. For C' our assumption leads us to $M \models \neg C'$. Furthermore, as the factorization-rule deletes doublicates of literals, we may assume that C' doesn't contain L. Hence according to our model construction L has to be interpreted as true yielding $M \models L$ and $M \models C' \vee L = C$.*
2. *For the induction step, let us assume that the lemma holds for all clauses $C' \in I_{DSL}(Inst(\mathcal{C}))$ with $np(C') \leq N$, $N \geq 0$. For $C \in I_{DSL}(Inst(\mathcal{C}))$ with $np(C) = N + 1$ we have $C = C' \vee L$, with $np(C') = N$, $L = \neg S(t)$ or $L = A$ where A is a non-sort atom.*
   If $M \models L$ we are through.
   If $M \not\models \neg L$ and L is self-complementary, i.e., $L = \neg S(t)$ with $sort(t) \leq S$, L can be discarded from C by the elimination rule yielding C'. Due to our induction hypothesis we have $M \models C'$ such that we obtain $M \models C' \vee L$.
   If L is not self-complementary, there has to be a clause $C'' \vee L'$ according to our construction of M with $M \models \neg C''$, $np(C'') = 0$, and where $L' = \neg A$, if L is a non-sort literal, and $L' = S'(t)$ with $S' \leq S$, if $L = \neg S(t)$. In any case, L and L' are complementary, and $C' \vee C''$ is a resolvent of $C' \vee L$ and $C'' \vee L'$. $C' \vee C'' \in I_{DSL}(Inst(\mathcal{C}))$ and $np(C' \vee C'') = np(C') + np(C'') = N + 0 = N$ yield $M \models C' \vee C''$, such that we get $M \models C'$ due to $M \models \neg C''$. Hence $M \models C' \vee L = C$.

We still have to show that M is a Σ-model.

1. *$M \models S(t)$, if $sort(t) \leq S$.*
   This holds trivially due to our construction of the model M.
2. *If $S_1 \leq S_2$, $M \models S_1(t)$ entails $M \models S_2(t)$.*
   If $S_1(t)$ is true due to $sort(t) \leq S_1$ this holds trivially for $S_2(t)$ too. If $S_1(t)$ is interpreted true due to a clause $C \vee S_3(t)$ with $M \models \neg C$ and $S_3 \leq S_1$ the literals in C occur before $S_1(t)$ in the enumeration. As $S_1(t)$ occurs in the enumeration before $S_2(t)$, this is also the case for the literals in C. Hence $S_2(t)$ is interpreted true too.
3. *If $S_1 \perp\!\!\!\perp S_2$, $M \models S_1(t)$ entails $M \models \neg S_2(t)$.*
   If $sort(t) \leq S_1$ we trivially have $\models \neg S_2(t)$.
   $M \models S_1(t)$ may hold too due to a clause $C \vee S_3(t)$ with $S_3 \leq S_1$ and $M \models \neg C$.
   Now let us suppose $M \models S_2(t)$:
   If $sort(t) \leq S_2$ $S_1(t)$ is self-complementary. This would lead to $C \in I_{DSL}(Inst(\mathcal{C}))$ and $M \models C$.
   If $M \models S_2(t)$ due to a clause $C' \vee S_4(t)$ with $S_4(t) \leq S_2(t)$ and $M \models \neg C'$, $S_4(t)$ is complementary to $S_3(t)$. Hence $C \vee C' \in I_{DSL}(Inst(\mathcal{C}))$ and we have again a contradiction due to $M \models C \vee C'$.

□

3.3.4 Lifting Refutations

Not all ground refutations can be lifted to the predicate logic level. The reason for this is that negative ground sort literals may have no counterpart on the predicate logic side such that inference steps using it cannot be reproduced there. The following example illustrates this problem.

Example 3 *Given $\Sigma = \langle \mathcal{S}, \mathcal{SC}, \mathcal{P}, \mathcal{F} \rangle$ with*
$\mathcal{S} = \{S_1, S_2\}$,
$\mathcal{SC} = \emptyset$,
$\mathcal{P} = \mathcal{P}^1 = \{P\}$ and
$\mathcal{F} = \mathcal{F}^0 \cup \mathcal{F}^1$ with $\mathcal{F}^0 = \{a : S_2\}$ and $\mathcal{F}^1 = \{f : S_1\}$.
The set containing the following clauses is unsatisfiable:

$$C_1 = \neg P(x : S_2), \quad C_2 = P(f(y : S_1)),$$
$$C_3 = S_2(f(a)), \quad C_4 = S_1(a).$$

From the ground instances
$C_1^{gr} = \neg P(f(a)) \vee \neg S_2(f(a))$ and
$C_2^{gr} = P(f(a)) \vee \neg S_1(a)$,
one gets together with C_3 and C_4 the following deduction of the empty clause (we indicate also the corresponding steps on the predicate logic level and underline the literals that are resolved upon. "r." stands for "resolution step with"):

C_1^{gr}	$\underline{\neg P(f(a))} \vee \neg S_2(f(a))$	C_1	$\underline{\neg P(x : S_2)}$
r. C_2^{gr}	$\underline{\neg S_1(a)} \vee \neg S_2(f(a))$	r. C_2	$\underline{\neg S_2(f(y : S_1))}$
r. C_4	$\underline{\neg S_2(f(a))}$	-	-
r. C_3	□	-	-

The resolution step with C_4 cannot be reproduced on the predicate logic side as $\neg S_1(a)$ is not an instance of $\neg S_2(f(y : S_1))$. However, if the resolution step with C_3 is executed first, the ground proof can be lifted:

C_1^{gr}	$\underline{\neg P(f(a))} \vee \neg S_2(f(a))$	C_1	$\underline{\neg P(x : S_2)}$
r. C_2^{gr}	$\underline{\neg S_1(a)} \vee \underline{\neg S_2(f(a))}$	r. C_2	$\underline{\neg S_2(f(y : S_1))}$
r. C_3	$\underline{\neg S_1(a)}$	r. C_3	$\underline{\neg S_1(a)}$
r. C_4	$\square$	R. C_4	$\square$

This example demonstrates that it is not sufficient to delay inference steps on negative sort literals until a clause contains no other literals[7]. But it gives us the clou for imposing the right restriction on the ground level deductions such that every deduction of the empty clause respecting this restriction can be lifted to the predicate logic level. The main problem is to avoid inference steps on ground sort literals that are created by subsequent inference steps on the predicate logic side.

From the resolution step of C_3 on $\neg S_2(f(y : S_1))$ which yields $\neg S_1(a)$ we see that on the ground side the argument of the corresponding sort literal $\neg S_2(f(a))$ contains a as a subterm, which is the argument of the ground literal that corresponds to the generated sort literal. Thus, if we perform a ground level inference step on a negative sort literal whose argument is not a subterm of any other negative sort literal, we know that this literal will not appear on the predicate logic side by subsequent deduction steps.

Now, what happens if the arguments of the sort literals are equal, like in $\neg S_1(a) \vee \neg S_2(a)$? This clause can be a ground instance of $\neg S_2(x : S_1)$. But this case doesn't cause any problems for lifting, even if the first inference step concerns $\neg S_1(a)$ since according to the reformulation case of the elimination rule $\neg S_2(x : S_1)$ can be transformed to $\neg S_1(y : S_2)$[8].

The restriction to be imposed on the ground deductions is clear now:

1. inference steps involving a negative sorted literal $\neg S(t)$ are executed only, if there are only negative sort literals in the clause,
2. further, if an inference step is executed on $\neg S_1(t_1)$ then there is no other literal $\neg S_2(t_2)$ in the same clause such that t_1 is a subterm of t_2.[9]

We have to show that this restriction is complete. Fortunately, this is easy, as it can be considered as a specific instance of the LOCK-resolution refinement [Boyer, 1971] which can be adapted to the Σ_{DSL}-ground-calculus.

The LOCK-refinement is defined as follows: Every literal appearance in a set of ground clauses is associated with an index. The lock refinement allows only inference steps (like resolution, factorization) on the literals with the lowest index in the parent clause(s). The index of the literals in the resultant clause is inherited from the literals in the parent clauses. Further we assume that factorization eliminates only the literal with the higher index ("merging low").

The completeness proof of the LOCK-refinement for ground resolution can be easily transferred to the Σ_{DSL}-ground calculus. It follows the lines of the proof in [Chang and Lee, 1973, , pages 124–125]. We denote by $I_{DSL-LOCK}(C)$ the smallest set containing C which is closed under the Σ_{DSL}-inference rules respecting the LOCK-refinement.

Lemma 3
If C_G is an unsatisfiable set of ground clauses then $\square \in I_{DSL-LOCK}(C_G)$.
Proof: *The theorem trivially holds if $\square \in C_G$. We assume therefore $\square \notin C_G$.*

The literal access parameter *$k(C_G)$ of C_G is defined as the difference between the total number of literal appearences in C_G and the number of clauses in C_G. We prove the theorem by induction on $k(C_G)$.*

If $k(C_G) = 0$, then C_G consists only of unit clauses. For unit claues we have $I_{DSL}(C_G) = I_{DSL-LOCK}(C_G)$, as lock refinement doesn't represent a restriction on unit clauses. Due to the completeness of the Σ_{DSL}-calculus for ground clauses (cf. Theorem 3) we get $\square \in I_{DSL-LOCK}(C_G)$, if C_G is unsatisfiable.

Let us suppose that the theorem holds for all sets of ground clauses C with $k(C) \leq N$, and that we have $k(C_G) = N+1$ for C_G. Let L be a literal in C_G that has the maximal index and $C = C' \vee L$ a clause it appears in. If C_G is unsatisfiable, then this is also the case for $C_G^1 = (C_G - C) \cup \{C'\}$ as well as $C_G^2 = (C_G - C) \cup \{L\}$. For these clause sets we have $\square \in I_{DSL-LOCK}(C_G^i)$, $i = 1,2$, due to $k(C_G^i) \leq N$.
Replacing in the deductions from C_G^1 C' by $C = C' \vee L$ leads to LOCK-deductions as L is the literal with the maximal index. Therefore we have
$L \in I_{DSL-LOCK}(C_G)$ or $\square \in I_{DSL-LOCK}(C_G)$ (L can be eliminated by factorization). In the second case we are done. If $L \in I_{DSL-LOCK}(C_G)$ then $C_G^2 \subseteq I_{DSL-LOCK}(C_G)$. Due to $\square \in I_{DSL-LOCK}(C_G^2)$ we also have $\square \in I_{DSL-LOCK}(I_{DSL-LOCK}(C_G)) = I_{DSL-LOCK}(C_G)$.
$\square$

For realizing the necessary restriction for liftable ground deductions we have to choose an indexing where a negative sort literal $\neg S(t)$ has a higher in-

[7] This has not been recognized neither in [Weidenbach and Ohlbach, 1990]) nor in [Beierle *et al.*, 1992].

[8] Actually, the situation is a bit more complex, as the reformulation case of the elimination rule cannot be applied arbitrarily, cf. proof of Lemma 4

[9] Cohn's restriction on the semantic trees [Cohn, 1992] has about the same effect.

dex than every literal being not a negative sort literal as well as every negative sort literal whose argument has t as a proper subterm.

Example 4 *The following indexing satisfies this property for the ground literals of the ground clauses in Example 3 (we indicate the index values as exponents of the literals):*

$\neg P(f(a))^1$, $P(f(a))^1$, $S_2(f(a))^1$, $S_1(a)^1$, $\neg S_2(f(a))^2$, $\neg S_1(a)^3$.
The second deduction of Example 3 is a LOCK-deduction with respect to this indexing.

Lemma 4 (Lifting Lemma)
Given an indexing $\mathcal{I}$ of the literals in $Inst(C)$ with the following properties:

- *negative sort literals have a higher index than all other literals;*
- *for negative sort literals $\neg S_1(t_1)$ and $\neg S_2(t_2)$, where t_1 is a proper subterm of t_2 we have $\mathcal{I}(\neg S_1(t_1)) > \mathcal{I}(\neg S_2(t_2))$*

Then, for every clause $R \in I_{DSL-LOCK}(Inst(C))$ there is a clause $K \in I_{DSL}(C)$ and a ground substitution σ such that $[\sigma]K \subseteq R$.
Proof: *We prove this lemma by induction on the definition of $I_{DSL-LOCK}(Inst(C))$.*

Due to the definition of $Inst(C)$ there is for every ground clause $R \in Inst(C))$ a clause $K \in C$ and a ground substitution σ with $[\sigma](K) = R$.

For the induction step be $R_1, R_2 \in I_{DSL-LOCK}(Inst(C))$ with $R_i = A_i \vee L_i$, i =1,2, for which there are $K_1, K_2 \in I_{DSL-LOCK}(C)$ and ground substitutions σ_1, σ_2 with $[\sigma_1](K_1) \subseteq R_1$ and $[\sigma_2](K_2) \subseteq R_2$. L_1 and L_2 have the lowest index in their clauses. Further, let R be obtained either from R_1 by an elimination or factorization step over L_1 or be obtained from R_1 and R_2 by a resolution step over L_1 and L_2; i.e., we have $R = A_1$ resp. $R = A_1 \vee A_2$.

First, we consider the case that the inference step on the ground side cannot be reproduced on the predicate logic side. Without loss of generality we may assume that the there is no literal $M \in K_1$ with $\sigma_1(M) = L_1$. If $L_1 \notin [\sigma_1](K_1)$ we have $[\sigma_1](K_1) \subseteq A_1$. Hence $[\sigma_1](K_1) \subseteq R$.
If $L_1 \in [\sigma_1](K_1)$, L_1 has to be in $\neg SL(\sigma_1)$; i.e., L_1 is a negative sort literal $\neg S(t)$ and we have $t \leftarrow x \in \sigma_1$ with $sort(x) = S$. Due to our indexing, A_1 as well as K_1 consist of negative sort literals only. Further, $x \in var(K_1)$ and our indexing entail that there is a sort literal $\neg S'(x)$ in K_1. There has to be a literal in K_1 containg x. If it would be of the form $\neg S''(t'(x))$ then $\neg\sigma_1(S''(t'(x))) = \neg S''(t'(t))$ would be in R_1. However, the argument of $\neg S''(t'(t))$ contains t as a proper subterm, such that it should have a lower index than $L_1 = \neg S(t)$. But L_1 is supposed to have the lowest one. Hence, we have $K_1 = B_1 \vee \neg S'_1(x) \vee ... \vee \neg S'_n(x)$ with $x \notin var(B_1)$ and $sort(x) = S$.
If there is a common subsort S^* of $S'_1, ..., S'_n$ and S then all these sort literals can be discarded by the elimination rule such that we have $B_1 \in I_{DSL}(C)$. With $\sigma'_1 = \sigma_1 - \{x \leftarrow t\}$ we get $[\sigma'_1]B_1 \subseteq A_1 \subseteq R$.
In the other case there is one S'_i with $S \not\leq S'_i$ (otherwise S would be such a common subsort). Let $B_2(x) = \neg S'_1(x) \vee ... \vee \neg S'_{i-1}(x) \vee \neg S'_{i+1}(x) \vee ... \vee S'_n(x)$ such that $K_1 = B_1 \vee B_2(x) \vee \neg S'_i(x)$. We have to distinguish the following two cases:

1. *$S'_i \leq S$:*
   Applying the elimination rule on $\neg S'_i(x)$ yields: $K'_1 = B_1 \vee B_2(y)$ with $y \notin var(B_1)$ and $sort(y) = S'_i$. For $\sigma''_1 = (\sigma_1 - \{x \leftarrow t\}) \cup \{y \leftarrow t\}$ we obtain $\neg SL(\sigma''_1) = (\neg SL(\sigma) - \{\neg S(t)\}) \cup \{\neg S'_i(t)\}$ and $\sigma''_1(K'_1) = \sigma_1(K_1) - \{\neg S'_i(t)\}$ such that $[\sigma''_1]K'_1 = [\sigma_1]K_1 - \{\neg S(t)\}$. Hence $[\sigma''_1]K'_1 \subseteq A_1 \subseteq R$.
2. *$S'_i \not\leq S$:*
   Here the reformulation case of the elimination rule is applicable, and we obtain: $K''_1 = B_1 \vee B_2(y) \vee \neg S(y)$ with $y \notin var(B_1)$ and $sort(y) = S'_i$.
   For σ''_1 (being defined as above) we have $\neg SL(\sigma''_1) \subseteq (\neg SL(\sigma_1) - \{\neg S(t)\}) \cup \{\neg S'_i(t)\}$ ($sort(t) \leq S'_i$ may hold or not) and $\sigma''_1(K''_1) = (\sigma_1(K_1) - \{\neg S'_i(t)\}) \cup \{\neg S(t)\}$ such that $[\sigma''_1]K''_1 \subseteq [\sigma_1]K_1$.
   As $\sigma''_1(\neg S(y)) = L_1$ the literal missing in K_1 appears now in K'_1.

Hence, we may now assume that $K_1 = B_1 \vee M_1$ and $K_2 = B_2 \vee M_2$ with $\sigma_1(M_1) = L_1$ and $\sigma_2(M_2) = L_2$. We consider now the different calculus rules that can be applied:

1. *$R = A_1 \vee A_2$ is a resolvent of R_1 and R_2; i.e., L_1 and L_2 are complementary.*
   We may assume that K_1 and K_2 as well as σ_1 and σ_2 have no variables in common. Therefore $\sigma = \sigma_1 \cup \sigma_2$ is a substitution and because of $\sigma(M_1) = L_1$ and $L_2 = \sigma(M_2)$ σ is a refuting substitution of M_1 and M_2. Due to Lemma 1 there is a maximally general refuting substitution Θ of M_1 and M_2 and a ground substitution ϱ such that $\Theta\varrho \supseteq \sigma$ and $\neg\varrho SL(\Theta) \cup \neg SL(\varrho) \subseteq \neg SL(\sigma)$.
   Hence we have for $K = \Theta(B_1 \vee B_2) \vee \neg SL(\Theta)$
   $[\varrho]K = \varrho\Theta(B_1 \vee B_2) \vee \neg\varrho SL(\Theta) \vee \neg SL(\varrho) \subseteq$
   $\sigma(B_1 \vee B_2) \vee \neg SL(\sigma) =$
   $\sigma_1(B_1) \vee \sigma_2(B_2) \vee \neg SL(\sigma_1) \vee \neg SL(\sigma_2) \subseteq$
   $A_1 \cup A_2 = R$.
2. *$R = A_1$ is obtained by the restricted version of factorization on L_1 in R_1;*
   i.e, we have $A_1 = A'_1 \vee L_1$.
   On the predicate logic side, we have to consider

the following cases:

(a) There is no literal $M \in B_1$ with $\sigma_1(M) = L_1$. Hence we have $[\sigma_1]K_1 \subseteq A'_1 \vee L_1 = R$.

(b) $B_1 = B'_1 \vee M'_1$ with $\sigma_1(M'_1) = L_1$; σ_1 is a refuting substitution of $\neg M_1$ and M'_1. Hence, there is a maximally general refuting substitution Θ of $\neg M_1$ and M'_1 and $K = \Theta(B_1) \vee \neg SL(\Theta)$ is a factor of B_1. Applying Lemma 1 again we get $\Theta\varrho \supseteq \sigma_1$ and $\neg\varrho SL(\Theta) \cup \neg SL(\varrho) \subseteq \neg SL(\sigma_1)$. Hence we obtain for K: $[\varrho]K = \varrho(\Theta(B_1)) \vee \neg\varrho SL(\Theta) \vee \neg SL(\varrho) \subseteq \sigma_1(B_1) \vee \neg SL(\sigma_1) = [\sigma_1]B_1 \subseteq A_1 = R$.

3. $R = A_1$ is obtained by an elimination step from $R_1 = A_1 \vee L_1$; i.e., L_1 is a self complementary sort literal. σ_1 is a refuting substitution of M_1. According to Lemma 1 there is a maximally general refuting substitution Θ of M_1 such that $K = \Theta(B_1) \vee \neg SL(\Theta)$ is the result of an elimination step, and we obtain analougously to above $[\varrho]K \subseteq R$.

□

The proper lifting theorem is now easy to prove:

Theorem 4 (Lifting of Refutations)
For any set of clauses C we have:
$\square \in I_{DSL}(Inst(C))$ *implies* $\square \in I_{DSL}(C)$.
Proof:

$\square \in I_{DSL}(Inst(C))$	*(LOCK-compl.)*	$\Longrightarrow$
$\square \in I_{DSL-LOCK}(Inst(C))$	*(lifting lemma)*	$\Longrightarrow$
$\square \in I_{DSL}(C)$		

□

3.4 On Refinements of the Calculus

In contrast to ordinary order-sorted logic, unification never fails here due to the incompatibility of sorts unless the sort of a variable and its replacing term are disjoint. In certain cases, this may lead to a prohibitive explosion of the search space, such that refinements putting additional conditions on the success of a unification will be useful.

An obvious restriction is that a unification should fail, if a generated sort literal is not refutable, i.e., if it can't be "resolved away". In this case, the generated clause can't contribute to the deduction of the empty clause. It is clear that determining whether a sort literal is refutable or not is undecidable in general. But we can establish sufficient conditions for the non-refutability of sort-literals.

Sort literals in $\neg SL(\Theta)$ are always negative. Hence, they can be made complementary with positive sort literals only. Such sort literals aren't generated dynamically. Therefore, a generated sort literal is not refutable, if it can't be made complementary with any positive sort literal of an input clause[10]. Note, the generated sort literals have not to be checked for self-complementarity. A sort literal $\neg S(t)$, where t is a "real" term, is self-complementary, if $sort(t) \leq S$. Hence, it will not be generated, if a variable $x : S$ is replaced by t. A sort literal $\neg S(y : S')$ with a variable y can be made self-complementary, if S and S' have a common subsort. However, it is easy to check that in this case there is another maximally general refuting substitution having the same effect where $\neg S(y : S')$ is not generated.[11]

As these positive sort literals are known beforehand this subset of non-refutable sort literals can be determined or characterized when compiling the set of input clauses. Such a characterization need not be complete. On the contrary, simple conditions have the advantage that they can be checked more efficiently than the more complex complete ones. E.g., if we look only at the sorts in the predicate position and disregard the arguments, we see that a sort literal $\neg S(t)$ is non-refutable if there is no clause $S'(t') \vee C'$ in the input set with $S' \leq S$.

Using such a refinement the computational behaviour of the Σ_{DSL}-calculus depends largely on the characteristics of the input clause set. There is an interesting case to consider. If no input clause contains a positive sort-literal then all generated sort literals are non-refutable. Hence unifications should succeed only, if no sort literals have to be generated. When looking at the conditions of Definition 5 we see that this holds if the sort of a replacing variable y is a greatest common subsort of the sorts of all the variables in $X = \{x \in domain(\Theta) \mid \sigma(x) = y\}$ being the variables y replaces in Θ. Otherwise, there would be a variable $x' \in X$ with $sort(y) \not\leq sort(x')$. Thus, if $\mathcal{SC}$ represents a lower semi-lattice, our refinement behaves exactly like order-sorted resolution [Walther, 1987].

Other refinements are more difficult to obtain. In particular, combining our calculus with the usual resolution refinements leads mostly to incompleteness. This is the case e.g. for the set-of-support strategy and all refinements that can be considered as a specialization of it, like linear resolution or model elimination.

The following simple example illustrates this:

Example 5 *Given $\Sigma = \langle \mathcal{S}, \mathcal{SC}, \mathcal{P}, \mathcal{F} \rangle$ with*
$\mathcal{S} = \{S_1, S_2\}$,
$\mathcal{SC} = \emptyset$,
$\mathcal{P} = \mathcal{P}^1 = \{P\}$,

[10] This corresponds to the "pure literal rule" of the connection graph procedure [Kowalski, 1975].

[11] This observation has the consequence that given an input clause set containing no negative sort literals we need for completeness only the reformulation case of the elimination rule together with the resolution and factorization rule.

$\mathcal{F} = \mathcal{F}^0 = \{a_1 : S_1\}$, and let
$C_1 = P(a_1)$, $C_2 = \neg P(x : S_2)$, $C_3 = S_2(a_1)$.
It is easy to check that $\{C_1, C_2, C_3\}$ is unsatisfiable. $\{C_3\}$ is a set of support as $\{C_2, C_3\}$ is satisfiable. However, no resolution step is possible neither between C_3 and C_1 nor between C_3 and C_2.

The example above shows that standard backward chaining procedures are insufficient for the Σ_{DSL}-logic, as well as for other logics of the non-substitutional framework. If one wants to be sure to preserve completeness one has to use the level saturation method, i.e., all possible resolvents have to be computed in a kind of forward chaining mode from a clause set.

4 Related Work

Cohn presents in [Cohn, 1992] a logic that can be considered as the most general one of the non-substitutional framework, as it allows for possibly empty sorts and imposes no restrictions on the axiomatization of the sort theory. However, it is not general enough such that our calculus can be considered as an instance of it.

When axiomatizing our sort constraints one gets for $S_1(x) \leq S_2(x)$ $\forall x : \top\ \neg S_1(x) \vee S_2(x)$ and for $S_3(x) \bot S_4(x)$ $\forall x : \top\ \neg S_3(x) \vee \neg S_4(x)$

The Σ_{DSL}-resolution rule is covered by Cohn's binary resolution rule for non-sort literals and his characteristic resolution rule for sort literals being characteristic literals in his terminology. Since DISJ-substitutions are sort consistent, no negated sort literals need to be generated for variables whose sort is possibly empty. Further, the residues stemming from the sort theory are always empty as sort literals can be made complementary without further preconditions, except those about the sort membership of a term.

The Σ_{DSL}-elimination rule isn't, however, an instance of Cohn's evaluation rule, as it is executed without applying substitutions. If a literal has to be instantiated, the instantiation has to be produced by a preceeding application of the two resolutions rules or the literal has to be resolved with itself. On the other hand, the inverse doesn't hold either. The evaluation rule allows to infer $C \vee S_2(t)$ from $C \vee S_1(t)$ if $S_1 \leq S_2$ holds, which is not possible by Σ_{DSL}-elimination. Therefore it is not possible to use Cohn's results directly for our logic, i.e., even though certain arguments are similar, the completeness proofs, in particular, have to be developed from scratch.

From a practical viewpoint, Cohn's calculus is difficult to implement. Lacking the notion of maximally general unifier/refuting substitution an implementation has to "guess" the appropriate instantiation. However, to be fair, one has acknowledge that it is very difficult to define this notion in such a general setting.

Comparing the Σ_{DSL}-logic to Cohn's earlier LLAMA system the main difference concerns the sort structure. LLAMA requires a complete boolean lattice being interpreted lattice theoretically, i.e., we have $U_{S_3} = U_{S_1} \cap U_{S_2}$ for the interpretation of the greatest lower bound S_3 of two sort S_1 and S_2. The disjointness of two sorts can result implicitly from the disjointness of the base sorts being located just above $\bot$. In our logic the sort hierarchy is interpreted order-theoretically, i.e., we have for the interpretation of a maximal lower bound $U_{S_3} \subseteq U_{S_1} \cap U_{S_2}$.

Further we impose no restrictions on the sort hierarchy. These would be of no value, as e.g. we still need multiple maximally general refuting substitutions if $\mathcal{S}$ wrt. $<$ is a lower semi-lattice. The following example demonstrates why this is the case:

Example 6 *Given $\Sigma = \langle \mathcal{S}, \mathcal{SC}, \mathcal{P}, \mathcal{F} \rangle$ with*
$\mathcal{S} = \{S_1, S_2, S_3\}$,
$\mathcal{SC} = \{S_3 < S_1, S_3 < S_2\}$,
$\mathcal{P} = \mathcal{P}^1 = \{P, Q\}$,
$\mathcal{F} = \mathcal{F}^0 = \{a_1 : S_1, a_2 : S_2, a_3 : S_3\}$, and let

$$C_1 = \neg P(x_1 : S_1) \vee Q(x_1), \quad C_2 = P(x_2 : S_2),$$
$$C_3 = \neg Q(a_3),$$
$$C_{41} = \neg Q(a_1), \quad C_{42} = S_2(a_1),$$
$$C_{51} = \neg Q(a_2), \quad C_{52} = S_1(a_2).$$

There are 3 maximally general refuting substitutions of $\neg P(x_1 : S_1)$ and $P(x_2 : S_2)$:
$\Theta_1 = \{x_1 \leftarrow y_1 : S_1, x_2 \leftarrow y_1\}$
with $\neg SL(\Theta_1) = \neg S_2(y_1 : S_1)$,
$\Theta_2 = \{x_1 \leftarrow y_2 : S_2, x_2 \leftarrow y_2\}$
with $\neg SL(\Theta_2) = \neg S_1(y_2 : S_2)$,
$\Theta_3 = \{x_3 \leftarrow y_3 : S_3, x_2 \leftarrow y_3\}$ *with* $\neg SL(\Theta_3) = \Box$.
With Θ_3 we can refute $C_3 = \{C_1, C_2, C_3\}$, but neither $C_1 = \{C_1, C_2, C_{41}, C_{42}\}$ nor $C_2 = \{C_1, C_2, C_{51}, C_{52}\}$. This is only possible with Θ_1 and Θ_2 respectively.

Thus in general, given a sort signature together with an arbitrary set of input clauses, it makes no difference for the computational behavior of our calculus, whether we have a sort-lattice or not.

If we want to have only one maximally general refuting substitution the sort hierarchy needs to be interpreted lattice-theoratically. With an appropriate extension of our calculus, Θ_3 will suffice for refuting C_1 and C_2. E.g., in the refutation of C_1 with Θ_3 the unit clause $\neg S_3(a_1)$ is produced. This literal is complementary under the condition that a_1 belongs also to the sort S_2. Therefore, applying the extended elimination rule should yield $\neg S_2(a_1)$, which is a sound inference under an lattice-theoratical interpretation due to $\forall x : \top\ S_3(x) \leftrightarrow S_1(x) \vee S_2(x)$. Deducing the empty clause from $\neg S_2(a_1)$ is now straightforward.

The DSPF-logic presented in [Weidenbach and Ohlbach, 1990] is quite similar to our logic. The two differ mainly in the interpretation of functions, which

are partial in the DSPF-logic and total in ours, and in the definitions of most general unifiers/maximally refuting substitutions. Further, the DSPF-logic allows only sort constraints specifying the subsort-relationship of two sorts. The partial interpretation of functions has consequences on the normalization procedure and demands an additional calculus rule for dealing with terms that are not defined. The most general unifiers of the DSPF-logic correspond to the classical ones, where no new variables are indroduced and old variables preserve their sort, such that one is sufficient for preserving completeness. However, given Θ_{DSPF}, a most general unifier of the DSPF-logic, and Θ_{DSL}, a corresponding maximally general refuting substitution of the Σ_{DSL}-logic, one can show that it is always possible to deduce $\Theta_{DSL}(C) \vee \neg SL(\Theta_{DSL})$ from $\Theta_{DSPF}(C) \vee \neg SL(\Theta_{DSPF})$ by elimination and factorization steps, i.e., the uniqueness of the most general unifiers does not lead to a reduction of the search space.

The two calculi can be combined in a straightforward way. For allowing partial functions in our logic we need only to add the calculus rule treating undefined terms and use the normalization procedure of the DSPF-logic. In this way, we get a calculus for the two logics.

5 Conclusion

Logics of the non-substitutional framework illustrate the classical tradeoff between expressiveness and efficiency, and the Σ_{DSL}-logic is not an exception. However, we think that our logic is a good basis for a knowledge representation language suiting for domains where it is insufficient to have statically declared sorts or types . The degrading effects on the spearch space can be limited, if the proposed refinement is used. The calculus is straightforward to implement, and for the cases, where the interpretation of the sort structure or the functions doesn't match the users needs, we have presented some modifications of our calculus.

References

[Beierle *et al.*, 1992] C. Beierle, U. Hedtstück, U. Pletat, J. Siekmann, and P.H. Schmitt. An order-sorted predicate logic for knowledge represention systems. *Artificial Intelligence*, 55(2-3):149–191, 1992.

[Boyer, 1971] R. S. Boyer. *Locking a Restriction of Resolution.* PhD thesis, University of Texas at Austin, 1971.

[Chang and Lee, 1973] C.-L. Chang and R.C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Computer Science and Applied Mathematics Series. Academic Press, New York, 1973.

[Cohn, 1987] A. G. Cohn. A more expressive formulation of many sorted logic. *Journal of Automated Reasoning*, 3:113–200, 1987.

[Cohn, 1989] A. G. Cohn. On the appearance of sortal literals: a non substitutional framework for hybrid reasoning. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 55–66, 1989.

[Cohn, 1992] A. G. Cohn. A many sported logic with possibly empty sorts. In D. Kapur, editor, *Automated Deduction - CADE-11, Proceedings*, volume 607 of *Lecture Notes in Artifical Intelligence*, pages 633–647. Springer-Verlag, 1992.

[Frisch, 1991] A. M. Frisch. The substitutional framework for sorted deduction: fundamental results on hybrid reasoning. *Artificial Intelligence*, 49:161–198, 1991.

[Kowalski, 1975] R. Kowalski. A proof procedue using connection graphs. *Journal of the Association for Computing Machinery*, 22(4):572–595, October 1975.

[Loveland, 1978] D. Loveland. *Automated Theorem Proving: A Logical Basis*, volume 6 of *Fundamental Studies in Computer Science*. North-Holland, New York, 1978.

[Oberschelp, 1962] A. Oberschelp. Untersuchungen zur mehrsortigen Quantorenlogik. *Mathematische Annalen*, 145:297–333, 1962.

[Schmidt-Schauß, 1989] M. Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*, volume 395 of *Lecture Notes in Artifical Intelligence*. Springer Verlag, 1989.

[Walther, 1987] C. Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation.* Research Notes in Artificial Intelligence. Pitman, London, and Morgan Kaufmann, Los Altos, Calif., 1987.

[Weidenbach and Ohlbach, 1990] Ch. Weidenbach and H.-J. Ohlbach. A resolution calculus with dynamic sort structures and partial functions. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 688–693, Stockholm, Sweden, 1990. Pitman Publishers.

Quantifier Elimination in Second–Order Predicate Logic

Dov Gabbay
Imperial College, Dept. of Computing
Queens Gate
London SWZ 2AZ, England

Hans Jürgen Ohlbach
Max–Planck–Institut für Informatik
Im Stadtwald
Saarbrücken 11, Germany *

Abstract

An algorithm is presented which eliminates second–order quantifiers over predicate variables in formulae of type $\exists P_1, \ldots, P_n \psi$ where ψ is an arbitrary formula of first–order predicate logic. The resulting formula is equivalent to the original formula – if the algorithm terminates. The algorithm can for example be applied to do interpolation, to eliminate the second–order quantifiers in circumscription, to compute the correlations between structures and power structures, to compute semantic properties corresponding to Hilbert axioms in non classical logics and to compute model theoretic semantics for new logics. An earlier version of the paper has been published in [GO92b].

Key Words: Quantifier Elimination, Second–Order Predicate Logic, Circumscription, Interpolation.

1 Introduction

Automatizing reasoning for second–order predicate logic is much more difficult than for first–order predicate logic. Procedures that transform a given second–order formula into an equivalent first–order formula are therefore extremely useful. This transformation is of course not always possible, but there are important problem classes where the second–order formulae have a characteristic structure which allows for a transformation into an equivalent first–order formula.

Several methods have been developed for computing from a given second–order formula an equivalent first–order formula. These methods basically fall into two classes. The first class of algorithms computes or guesses suitable instantiations for the second–order predicate variables that are guaranteed to preserve equivalence [Ack35a, vB84, Sza92, Sim92]. The idea of the second class of algorithms is to compute sufficiently many consequences from the formulae containing the second–order variables and then keeping from the resulting set of formulae only those without the second–order variables [Ack35a, KK66, BGW92]. The algorithm we are going to present falls into this second class.

The structure of the formulae our algorithm can handle is $\exists P_1, \ldots, P_n \psi$ where the P_i are predicate variables for n–place predicates and ψ is an arbitrary first–order formula. Our algorithm essentially normalizes ψ into clause form and generates all (constraint–) resolvents of the clauses with the predicates P_i. It is shown that the subset of the generated clauses not containing predicates P_i (which may be infinite) is equivalent to the original formula. Since $\forall P_1, \ldots, P_n \psi \Leftrightarrow \neg \exists P_1, \ldots, P_n \neg \psi$ ($\Leftrightarrow$ is the equivalence sign), the algorithm can of course also handle universal quantifiers by reducing this case to the case with existential quantifiers. This algorithm is simple and it can be realized easily with existing theorem provers, for example OTTER.

Let us illustrate the SCAN algorithm[1] with some simple examples.

It is easy to see that

$$\exists P \begin{pmatrix} P \vee Q \\ \neg P \vee R \end{pmatrix} \text{ is logically equivalent to } Q \vee R$$

where $Q \vee R$ is just the resolvent between the two clauses on the left hand side. The "$\Rightarrow$"–direction follows from the fact that $Q \vee R$ as a resolvent is of

*This work was supported by the ESPRIT project 3125 MEDLAR and by the "Sonderforschungsbereich" 314, "Künstliche Intelligenz und wissensbasierte Systeme" of the German Research Council (DFG). The first author is a SERC Senior Research Fellow.

[1] SCAN means 'Synthesizing Correspondence Axioms for Normal logics.' We developed this algorithm for a particular application and the name was chosen before we realized that it is applicable in a general context.The casual chain of the papers is the following: First SCAN was introduced in [GaOh92a]. This stimulated [Sza92]. Both papers then stimulated [Sim92].

course implied by the original formula. To see the "⇐"-direction, suppose Q is true in an interpretation. In this case the assigment for P, which is existentially quantified, can be chosen to be true also, making the existentially quantified formula true as a whole. If instead, R is true then P must be chosen to be false and again the left hand side formula is true. Thus, the existentially quantified P can be eliminated by just taking the single resolvent with P.

A slightly more complex example illustrates that in fact *all* (not redundant) resolvents with the second-order predicate have to be generated.

$$\exists P \begin{pmatrix} P \vee Q \\ \neg P \vee R \\ \neg P \vee S \end{pmatrix} \text{ is logically equivalent to } \begin{pmatrix} Q \vee R \\ Q \vee S \end{pmatrix}$$

If Q is false in a model, it is necessary that R and S are both true in order to choose P such that all three clauses on the left hand side become true. Falsity of Q enforces truth of both R and S on the right hand side only if both resolvents are present.

In the presence of second-order predicates with arguments, the resolution rule has to be changed slightly, as the third example demonstrates.

$$\exists P \begin{pmatrix} P(a) \vee Q \\ \neg P(b) \vee R \end{pmatrix} \text{ is logically equivalent to } a = b \Rightarrow (Q \vee R)$$

Only in models of the right hand side where a and b are mapped to the same objects, it is necessary that one of Q or R must be true in order to satisfy the left hand side. If a and b are interpreted differently, we may well choose both $P(a)$ and $\neg P(b)$ to be true. That means instead of unification, just a constraint for the arguments of the resolution literals has to be generated.

Further examples are given in the applications section. The SCAN algorithm is defined in the next section and its soundness is proved. In the third section we give a more detailed comparison with other approaches. Finally various applications of quantifier elimination are discussed. We conclude with a section on limitations of the algorithm and some implementation hints.

2 The SCAN Algorithm

The algorithm is defined and its soundness is proved.

Definition 2.1 (The SCAN Algorithm)
Input to SCAN is a formula $\alpha = \exists P_1, \ldots, P_n\ \psi$ with predicate variables $P_1, \ldots, P_n$ and an arbitrary first-order formula ψ.
Output of the SCAN — if it terminates — is a formula φ_α which is logically equivalent to α, but not containing the predicate variables $P_1, \ldots, P_n$.

SCAN performs the following three steps:

1. ψ is transformed into clause form using second order skolemization. That means the resulting formula has the form: $\exists P_1, \ldots, P_n \exists f_1, \ldots, f_n \psi'$ where the f_i are the Skolem functions and ψ' is a set of clauses. From the algorithm's point of view, the quantifier prefix can be ignored. Therefore ψ' is treated as an ordinary clause set with the usual Skolem constants and functions.

2. All C-resolvents and C-factors with the predicate variables $P_1, \ldots, P_n$ have to be generated. C-resolution ('C' for constraint) is defined as follows:

$$\frac{\begin{array}{ll} P(s_1, \ldots, s_n) \vee C & P(\ldots) \text{ and } \neg P(\ldots) \\ \neg P(t_1, \ldots, t_n) \vee D & \text{are the } \textit{resolution literals} \end{array}}{C \vee D \vee s_1 \neq t_1 \vee \ldots \vee s_n \neq t_n}$$

and the C-factorization rule is defined analogously:

$$\frac{P(s_1, \ldots, s_n) \vee P(t_1, \ldots, t_n) \vee C}{P(s_1, \ldots, s_n) \vee C \vee s_1 \neq t_1 \vee \ldots \vee s_n \neq t_n}.$$

Notice that only C-resolutions between different clauses are allowed (no self resolution). A C-resolution or C-factorization can be optimized by destructively resolving literals $x \neq t$ where the variable x does not occur in t with the reflexivity equation. C-resolution and C-factorization takes into account that second order quantifiers may well impose conditions on the interpretations which must be formulated in terms of equations and inequations.

As soon as *all* resolvents and factors between a particular literal and the rest of the clause set have been generated (the literal is 'resolved away'), the clause containing this literal must be deleted (purity deletion). If all clauses are deleted this way, this means that α is a tautology.

All equivalence preserving simplifications may be applied freely. These are for example:

- Tautologous resolvents can be deleted.
- Subsumed clauses can be deleted.
- Subsumption factoring can be performed. Subsumption factoring means that a factor subsumes its parent clause. This may be realized by just deleting some literals. For example $Q(x), Q(a)$, where x is a variable, can be simplified to $Q(a)$.
- Subsumption resolution can also be performed. Subsumption resolution means that a resolvent subsumes its parent clause, and this again may be realized by deleting some literals [OS91] (see also example 4.4). For example the resolvent between P, Q and $\neg P, Q, R$ is just Q, R such that $\neg P$ can be deleted from the clause. (An instance of this operation is realized as so

called 'unit_deletion' in the OTTER theorem prover.)

If an empty clause is generated, this means that α is contradictory.

3. If the previous step terminates and there are still clauses left then reverse the skolemization. A method for reversing the skolemization in a set F of clauses is (1) to abstract all arguments of all occurrences of Skolem functions by variables, i.e. $f(s_1, \ldots, s_n)$ is replaced with $f(x_1, \ldots, x_n)$ and additional literals $x_i \neq s_i$ are added to the clause where the x_i are fresh variables and (2) to consistently rename all variables such that the arguments of all occurrences of the Skolem function are the same. If this is possible and $F[f(x_1, \ldots, x_n)]$ is the result then $\forall x_1, \ldots, x_n \exists y\, F[y]$ is the solution. This process is repeated for all Skolem functions.

   If it is not possible to rename the variables consistently, the only chance is to take parallel Henkin quantifiers [Hen61] (see example 4.5) or leave the second–order quantification.

◁

The next example illustrates the different steps of the SCAN algorithm in detail. The input is:
$\exists P\ \forall x, y\ \exists z\ (\neg P(a) \vee Q(x)) \wedge (P(y) \vee Q(a)) \wedge P(z)$.
In the first step the clause form is to be computed:

$$\begin{array}{ll} C_1 & \neg P(a), Q(x)) \\ C_2 & P(y), Q(a) \\ C_3 & P(f(x,y)) \end{array}$$

f is a Skolem function. The second–order quantifier prefix is therefore $\exists P\ \exists f\ \forall x, y$. But this is only needed for the correctness proof below.

In the second step of SCAN we begin by choosing $\neg P(a)$ to be resolved away. The resolvent between C_1 and C_2 is $C_4 = Q(x), Q(a)$ which is equivalent to $Q(a)$ (this is one of the equivalence preserving simplifications). The C-resolvent between C_1 and C_3 is $C_5 = (a \neq f(x,y), Q(x))$. There are no more resolvents with $\neg P(a)$. Therefore C_1 is deleted. We are left with the clauses

$$\begin{array}{ll} C_2 & P(y), Q(a) \\ C_3 & P(f(x,y)) \\ C_4 & Q(a) \\ C_5 & a \neq f(x,y), Q(x) \end{array}$$

Selecting the next two P-literals to be resolved away yields no new resolvents. Thus, C_2 and C_3 are simply to be deleted as well. All P-literals have now been eliminated. Restoring the quantifiers we then get

$$\forall x\ \exists z\ Q(a) \wedge (a \neq z \vee Q(x))$$

as the final result (y is no longer needed.)

Theorem 2.2 (Correctness of SCAN)
If SCAN terminates for a formula α then α is logically equivalent to $SCAN(\alpha)$

Proof: The formulae under consideration contain a prefix of second–order existential quantifiers over predicate and function variables as the only second–order component. In order to prove the equivalence we can therefore take a standard first–order Tarskian model theory augmented with assignments of n-ary relations to n-place predicate variables and n-ary functions to n-place function variables.

Since we use second order skolemization, clause form generation as well as reversing the skolemization are equivalence preserving. Adding a resolvent or a factor to a clause set is also equivalence preserving. Therefore the only critical step in the SCAN algorithm is the purity deletion rule.

Removing a clause cannot make (in an interpretation) true clause sets false. Therefore every interpretation satisfying the clause set before the deletion satisfies it also after the deletion.

What we are left with to prove is that an interpretation satisfying the clause set without the pure clause also satisfies the clause set with the pure clause. And this turns out to be the really hard part of the proof where we have to exploit the second order character of the problem. What has to be exploited is that the predicate P is existentially quantified and therefore its interpretation can be chosen appropriately.

Before we come to the proof for the general case, it is useful to make some conceptual simplifications.

- Exploiting the equivalence $(P(s_1, \ldots, s_n) \vee C) \Leftrightarrow (P(x_1, \ldots, x_n) \vee C \vee x_1 \neq s_1 \vee \ldots \vee x_n \neq s_n)$ it can be assumed that the predicate P which has been "resolved away" in the pure clause C has only variables as arguments.
- The proof for an n-place predicate is not different to the proof for a one–place predicate. Just read x in $P(x)$ as a vector of variables. W.l.o.g we assume therefore that P is a one–place predicate.
- Since purity deletion is done after all resolvents and factors with the pure literal are generated, it can be assumed w.l.o.g that there is only one resolution partner in the clause set. If there are n resolution partners in the clause set then all proof steps below can be repeated n times.
- The clauses containing no resolution partners do not contain complementary literals with the predicate P. They are not touched during the purity deletion process. In the sequel they can therefore be ignored.
- The variables in the clauses can be renamed such that different clauses *share* the same variables.

There are two cases which have to be distinguished. The first case is that the predicate P occurs in the pure clause C only with one sign, either positively or negatively. The second case is that P occurs with both signs, i.e. C is self resolving. This is the case where *SCAN* may loop.

Let us now consider the first case and w.l.o.g assume the predicate P occurs only positively in C.

Thus, the situation before and after purity deletion looks as follows:

$$K = \left\{ \begin{array}{l} P(x), C(x) \qquad (=_{def} \ A) \\ Factors(A) \\ \hline \neg P(x_1), \ldots, \neg P(x_n), D(x_1, \ldots, x_n) \\ \hline S(x_1), \ldots, S(x_n), D(x_1, \ldots, x_n) \\ \quad \vdots \end{array} \right.$$

$$\rightarrow \ K' = \left\{ \begin{array}{l} Factors(A) \\ \hline \neg P(x_1), \ldots, \neg P(x_n), D(x_1, \ldots, x_n) \\ \hline S(x_1), \ldots, S(x_n), D(x_1, \ldots, x_n) \\ \quad \vdots \end{array} \right.$$

where C and D denote the remaining literals. These literals may well contain additional variables. For the purpose of this proof these variables can be ignored. C and D may also contain additional *positive* literals with the predicate symbol P. $S(x_1), \ldots, S(x_n), D(x_1, \ldots, x_n)$ stands for the $2^n - 1$ resolvents which are possible between these two clauses. S denotes either $\neg P$ or C. For example if $n = 2$ there are three resolvents:

$$\begin{array}{l} \neg P(x_1), \ C(x_2), D(x_1, x_2) \\ C(x_1), \neg P(x_2), D(x_1, x_2) \\ C(x_1), \ C(x_2), D(x_1, x_2) \end{array}$$

If tautologies are automatically eliminated those resolvents which are either themselves tautologies or which are derived from tautologies are not present. We shall see that the factors of A are needed to take over the role of those clauses which are derived from tautologies. Subsumed clauses, however, may be deleted without any effect on the proof.

Take any interpretation $\Im$ satisfying K' and mapping the symbol P to a predicate $\mathcal{P}$ and the variable x to a domain element a. If $\Im$ satisfies $C(x)$ or $P(x)$ then $\Im$ satisfies K and we are done. If $\Im$ falsifies both we move to an interpretation $\Im'$ by changing the assignment of P to a predicate $\mathcal{P}'$ which is like $\mathcal{P}$ except that $\mathcal{P}'(a)$ is true. Then $\Im'$ satisfies $P(x), C(x)$. We have to show that $\Im'$ still satisfies all the other clauses.

Assume $\Im'$ maps the variables x_i to some a_i. Let J be the set of variables which are mapped to a, i.e. $x_i \in J$ if $a_i = a$. For these variables, the truth value of $\neg P(x_i)$ has changed from *true* under $\Im$ to *false* under $\Im'$. For all other variables nothing has changed. Therefore if $\neg P(x_j), j \notin J$ is true under $\Im$, it is still true under $\Im'$. In this case all clauses containing this literal are still true. Now suppose $\neg P(x_j), j \notin J$ are all false under $\Im$. If for simplicity we assume $J = \{x_1, \ldots, x_j\}$, there is a clause $M = C(x_1), \ldots, C(x_j), \neg P(x_{j+1}), \ldots, \neg P(x_n), D(x_1, \ldots, x_n)$ among the resolvents. In this clause, all literals with predicate C and $\neg P$ are false under $\Im$. Since $\Im$ satisfies K', it must satisfy $D(x_1, \ldots, x_n)$. The interpretation of D has not changed. Therefore $\Im'$ satisfies $D(x_1, \ldots, x_n)$ as well. Thus, $\Im'$ satisfies all clauses in K.

It remains to be shown that in case of automatic tautology deletion the critical clause M is ***not*** deleted. If M would either itself be a tautology or derived from a tautology, the clause A would look like $A = P(x), P(y), C'(x, y)$. In this case the structure of M would be
$M = P(y_1), C'(x_1, y_1), \ldots, P(y_1), C'(x_j, y_j), \neg P(x_{j+1}), \ldots, \neg P(x_n), D(x_1, \ldots, x_n)$. That means for example $P(y_1), C'(x_1, y_1)$ would be false for all assignments of y_1, in particular for $\Im(y_1) = a$. This assigment would also falsify the factor $P(x), C'(x, x)$ of clause A, which cannot be the case[2].

From this we conclude that both $\Im'$ and $\Im$ satisfy $\exists P \ K$.

The remaining case to be considered is the case where the clause C is self resolving. Schematically the situation looks as follows:

$$K = \left\{ \begin{array}{l} P(x), \neg P(y), C(x, y) \\ \neg P(x), D(x) \\ \hline \neg P(y), D(x), C(x, y) \\ \neg P(y), D(x), C(x, x'), C(x', y) \\ \quad \vdots \ \text{(possibly infinitely many resolvents)} \end{array} \right.$$

$$\rightarrow \ K' = \left\{ \begin{array}{l} \neg P(x), D(x) \\ \hline \neg P(y), D(x), C(x, y) \\ \neg P(y), D(x), C(x, x'), C(x', y) \\ \quad \vdots \end{array} \right.$$

To simplify things, let us assume, neither $C(x, y)$ nor $D(x)$ contain negative occurrences of P. If in a given interpretation $\Im$ which maps x to some a_0, $D(x)$ is true, we can choose $P(x)$ to be true without further conflicts. If $D(x)$ is false, but $C(x, y)$ is true, where y is mapped to some a_1, $\Im$ satisfies K. If both $D(x)$ and $C(x, y)$ are false then the first resolvent enforces $\neg P(y)$ to be true. That means, $P(a_0)$ to be false enforces $P(a_1)$ to be false and therefore it has to be checked whether the first clause still remains true under the

[2]This argument does not imply that only binary factors are needed. Before the factor itself is operated on, its factor has to be generated. That means all factors are needed.

assignment $x \mapsto a_1$ etc. With the same arguments as in the base case this is proved with induction on n using the n^{th} resolvent. That means that in this case $\Im$ again satisfies $\exists P\ K$.

The case that $C(x)$ contains further negative literals with P means that there is another recursion loop which generates new branches of resolvents. Induction on the number of these further literals proves the statement.

The case that $D(x,y)$ also contains negative literals with P requires the integration of the arguments we used to prove the first case into the proof for the second case. This is technically complicated, but there are no further proof ideas needed.

Collecting everything together we can finally conclude $\alpha \Leftrightarrow SCAN(\alpha)$. ◁

If the formula given to SCAN contains a cycle in the P-literals, SCAN may keep on producing infinitely many clauses. In some cases the size of the clauses remains finite. According to a result of Ackermann [Ack35a, Ack35b] which can be adapted to our case, the (possibly infinite) conjunction of the P-literal free clauses is a solution. It is one of the advantages of SCAN that in this case a subsumption test on the resolvents may terminate the process and thus compute a finite result whereas otherwise only infinite junk would be produced.

As an example where this happens, apply SCAN to the formula:

$$\exists P\ \forall x,y\ (P(x,y) \Rightarrow P(y,x)) \wedge (P(x,y) \Leftrightarrow Q(x,y)).$$

Since the symmetry clause $\neg P(x,y), P(y,x)$ contains only a trivial cycle which can easily be recognized, SCAN stops and returns as expected the symmetry of Q.

There is, however, no proof that SCAN stops in all cases where there is a finite solution. If this were the case then it would be decidable whether a theorem $\exists x\ P(x)$ has finitely many different proofs or not.

As mentioned in the introduction, formulae with universal quantifiers have to be negated before giving them to SCAN and the result has to be negated again. The question may arise whether there is a possibility to treat universally quantified variables directly. Eliminating P from formulae $\forall P\ F[P]$ in some sense means factoring out the tautological part of $F[P]$. For example $(\forall P\ (P \vee \neg P) \wedge Q)$ is equivalent to Q, i.e. the P-part is tautologous. SCAN uses resolution as the basic operation, and resolution is sensitive to contradictions and not to tautologies. Therefore it is resolution which requires negation of the formula and elimination of the contradictious part.

In applications like circumscription, the structure of the formulae is $\forall P^*\ Q[P^*] \Rightarrow R[P^*]$ with a large $Q[P^*]$ and a small $R[P^*]$. In this case it is much more convenient to negate the formula yielding $\exists P^*\ Q[P^*] \wedge \neg R[P^*]$ because the big $Q[P^*]$ remains untouched.

There is a corollary derived from the proof of SCAN (2.2) which may be of some interest. The proof says that deleting a clause as soon as one literal is resolved away preserves equivalence. This may be exploited to eliminate only certain unwanted formulae. As an example, consider a PROLOG program containing a binary predicate P which is *symmetric*. Adding the symmetry clause to the program clauses PROLOG to loop. The corollary allows to eliminate the symmetry clause by generating all non redundant resolvents with the other PROLOG clauses. That means in this case that all clauses containing some $P(s,t)$ are duplicated with $P(t,s)$ replacing $P(s,t)$ in the copy. For all queries not containing P, the new PROLOG program is equivalent to the old one together with the symmetry clause.

3 Comparison with other Methods

Wilhelm Ackermann gave two procedures for eliminating existential quantifiers. Both eliminate only one quantifier at a time. The first one requires to bring the formula into a form

$$\exists P\ \forall x\ (A(x) \vee P(x)) \wedge \Delta[\neg P]$$

where $\Delta[\neg P]$ is a formula containing only negative occurrences of $P(x)$. The result is then $\Delta[A]$, i.e. all occurrences of $\neg P(x)$ are replaced with $A(x)$ in Δ. This method has difficulties in handling problems with clauses containing several occurrences of P. For example

$$\exists P\ \forall x,y \quad \begin{array}{l}(P(x,a) \vee P(a,x) \vee C(x)) \wedge \\ (\neg P(y,a) \vee \neg P(a,y) \vee D(y))\end{array}$$

falls into this problematic class. The SCAN–solution for this case, however, is simply $C(a) \vee D(a)$.

In its kernel the second method of Ackermann is actually quite similar to SCAN. Although his notation is very different to ours, it amounts to generating the conjunction of all P–free resolvents[3].

[3] Historical note: On page 401 of [Ack35a] there is the definition an operation

$$\mathcal{A}^{x_1,\ldots,x_n,z}_{y_1,\ldots,y_m} \wedge \mathcal{B}^{p_1,\ldots,p_n}_{q_1,\ldots,q_l,z} \rightarrow \mathcal{A}^{x_1,\ldots,x_n}_{y_1,\ldots,y_m} \vee \mathcal{B}^{p_1,\ldots,p_n}_{q_1,\ldots,q_l}$$

where the subscripts y_i stand for $P(y_i)$ and the superscripts x_i stand for $\neg P(x_i)$ in a clause containing also literals $\mathcal{A}$ or $\mathcal{B}$ respectively. Thus, contraction on z actually means resolution between $P(z)$ and $\neg P(z)$. The step to full resolution as we know it now is not that big.

It is, however, also restricted to one–place predicates. Literals $P(s_1, \ldots, s_n)$ have therefore to be written as $First(x, s_1) \wedge \ldots \wedge nth(x, s_n) \Rightarrow P'(x)$ before this method can be applied. This transformation blows up the formulae considerably. In fact, if SCAN is reduced to formulae with one-place predicates where all arguments are variables and no further simplifications of the resolvents are applied, you obtain Ackermann's second method.

Since Ackermann does not use resolution as we do, it is very difficult to integrate optimization steps like subsumption deletion etc. That means that his method would not terminate for the above example with the symmetry clause. Therefore SCAN is much easier to handle and it behaves better than Ackermann's method in such pathological cases.

The idea of generating consequences of the formulae with P and then taking the subset of P–free formulae is actually the kernel of other approaches to this problem. For example a theorem in [KK66] says that it is the set of *all* consequences you have to take. This is of course too much to be of practical value. A minimal subset free of redundancies should be sufficient. We showed that the set of resolvents without tautologies and subsumed clauses is sufficient. Bachmair, Ganzinger and Waldmann [BGW92] have gone even one stop further. Their "hierarchical theorem proving" approach allows the formulation of redundancy criteria based on term orderings. Furthermore they have incorporated equality reasoning by superposition principles. This mechanism can be used to get rid of existentially quantified predicate *and function symbols.*

4 Applications of Quantifier Elimination

As mentioned in the introduction, the applications we have in mind are classes of problems where formulae with the structure $\exists P_1, \ldots, P_n \psi$ or $\forall P_1, \ldots, P_n \psi$ occur. This need not be second–order formulae in the first place. Even in standard first–order logic there might be useful applications. Suppose there is an axiomatization of something in terms of a predicate P and maybe some other predicates, and by some reason it is known that only theorems not containing P are to be proved from these axioms. That means

$$\begin{array}{lll} & Axioms(P) & \Rightarrow Theorem \\ \text{iff} & \forall P\ (Axioms(P) & \Rightarrow Theorem) \\ \text{iff} & (\exists P\ Axioms(P)) & \Rightarrow Theorem \\ \text{iff} & SCAN(\exists P\ Axioms(P)) & \Rightarrow Theorem \end{array}$$

i.e. SCAN can optimize the axioms with respect to the particular class of theorems not containing P.

Actually the situation is a special case of *interpolation.* We have

$$\begin{array}{lll} & \forall Q, R\ \varphi(Q,P) & \Rightarrow \psi(P,R) \\ \text{iff} & (\exists Q\ \varphi(Q,P)) & \Rightarrow \forall R\ \psi(P,R) \\ \text{iff} & (SCAN(\exists Q\ \varphi(Q,P)) & \Rightarrow \neg SCAN(\exists R \neg \psi(P,R)) \\ \text{iff} & \varphi'(P) & \Rightarrow \psi'(P) \end{array}$$

i.e. SCAN does interpolation.

4.1 Circumscription

Circumscription is a transformation of formulae proposed by John McCarthy [McC80] for the purpose of formalizing non–monotonic aspects of commonsense reasoning. 'Circumscribing' a predicate means in semantic terms minimizing the extension of that predicate.

If for example the formula $Bird(Tweety)$ is circumscribed with respect to the predicate $Bird$, we want to formalize that $Tweety$ is the only bird at all, i.e. $Circ(Bird(Tweety), Bird) \Leftrightarrow$
$Bird(Tweety) \wedge \forall x\ Bird(x) \Rightarrow x = Tweety$.

Analogously $Circ(\forall x\ Bird(x) \Rightarrow fly(x), fly)$
$\Leftrightarrow \forall x\ Bird(x) \Leftrightarrow fly(x)$, i.e. if birds fly then minimizing the extension of fly adds the information that *only* birds fly.

Unfortunately circumscription in its general definition requires a second–order quantifier. For circumscribing a single unary predicate at a time, the definition is as follows:

$$\begin{array}{l} Circ(F(P), P) = \\ \quad F(P) \wedge \forall P^*\ (F(P^*) \wedge \forall y\ P^*(y) \Rightarrow P(y)) \\ \qquad\qquad \Rightarrow (\forall x\ P(x) \Rightarrow P^*(x)). \end{array}$$

So far only in special cases equivalent first–order formulae could be computed [Lif85]. SCAN offers a uniform way to compute first–order circumscriptions in most of the cases where there is one at all.

Example 4.1 (For Circumscription)
$Circ(P(a), P) = P(a) \wedge \forall P^*\ (P^*(a) \wedge$
$\forall y\ P^*(y) \Rightarrow P(y)) \Rightarrow (\forall x\ P(x) \Rightarrow P^*(x))$.
In order to get rid of the second–order quantification, the $\forall P^* \ldots$–formula is negated to obtain a version with existential quantifier $\exists P^* \ldots$ which is a suitable input to SCAN. Clausifying the negated formula yields

$$\begin{array}{lll} \exists P^*\ \exists x\ \forall y & & P^*(a) \\ & & \neg P^*(y), P(y) \\ & & P(x) \\ & & \neg P^*(x) \end{array}$$

Fortunately there is no Skolem function. Therefore we need not worry about skolemization and unskolemization. We just keep in mind that x is existentially quantified and therefore to be treated as a constant symbol.
Resolving P^* away yields

$$\begin{array}{ll} \exists x & P(a) \\ & P(x) \\ & x \neq a \end{array}$$

The result has to be negated again such that the final version is as expected:

$$Circ(P(a), P) \Leftrightarrow P(a) \wedge \forall x\ P(x) \Rightarrow x = a.$$

◁

Example 4.2 (For Circumscription)
$Circ(\forall x\ P(x) \Rightarrow Q(x), Q) =$
$\forall x\ P(x) \Rightarrow Q(x) \wedge \forall Q^*\ (\forall y\ P(y) \Rightarrow Q^*(y) \wedge$
$\forall y\ Q^*(y) \Rightarrow Q(y)) \Rightarrow (\forall x\ Q(x) \Rightarrow Q^*(x))$.

Negation of the $\forall P^* \ldots$-formula and clausification yields

$$\begin{array}{lll} \exists Q^*\ \exists x\ \forall y & \neg P(y), Q^*(y) \\ & \neg Q^*(y), Q(y) \\ & Q(x) \\ & \neg Q^*(x) \end{array}$$

Resolving Q^* away yields

$$\begin{array}{lll} \exists x\ \forall y & \neg P(x) \\ & Q(x) \\ & \neg P(y), Q(y) \end{array}$$

Negation again yields:

$$(\forall y\ P(y) \Rightarrow Q(y)) \Rightarrow (\forall x\ Q(x) \Rightarrow P(x))$$

Together with $\forall y\ P(y) \Rightarrow Q(y)$ this simplifies to $\forall y\ P(y) \Leftrightarrow Q(y)$. ◁

4.2 Power Structures

An *algebra* is a set together with some functions operating on this set. A *structure* is an algebra plus some additional relations between the elements of the algebra's carrier set. A *power structure* of a structure is again a structure. Its carrier set is just the powerset of the structure's carrier set and its functions and relations are obtained by lifting the structure's functions and relations to operate on sets instead of elements [Bri92]. For example a binary relation R can be lifted to a one-place function F:

$$F(X) = \{z \mid \exists x \in X\ R(x, z)\}$$

The *duality problem* is now to find the correspondences between the properties of the relation and the properties of the lifted function. For example transitivity of the binary relation corresponds to the property $F(F(X)) \subseteq F(X)$. Quite a number of theories in mathematics, logic, and computer science turn out to be instances of power structure constructions, duality theory, or correspondence theory respectively, in logic, or power domains in denotational semantics of non deterministic programs, just to name two of them.

With SCAN it is possible to do one direction and compute from the properties of the lifted function the properties of the underlying relation. For example in order to obtain transitivity from $F(F(X)) \subseteq F(X)$ we proceed as follows:
Applying the above definition of F in terms of R,
$\forall X\ \forall z\ z \in F(F(X)) \Rightarrow z \in F(X)$
is rewritten to
$\forall X\ (\exists x\ x \in F(X) \wedge R(x, z))$
$\Rightarrow (\exists x\ x \in X \wedge R(x, z))$
and further to
$\forall X\ \forall z\ (\exists x\ (\exists x'\ x' \in X \wedge R(x', x)) \wedge R(x, z))$
$\Rightarrow (\exists x\ x \in X \wedge R(x, z))$
which can also be written as
$\forall X\ \forall z\ (\exists x\ (\exists x'\ X(x) \wedge R(x', x)) \wedge R(x, z))$
$\Rightarrow (\exists x\ X(x) \wedge R(x, z))$
and this is now a typical quantifier elimination problem. Negation, application of SCAN and negation again yields the transitivity of R (c.f. example 4.3).

The other direction, computing from the properties of the structure the properties of the power structure, turns out to be a formula synthesizing problem that can be solved with resolution theorem provers by constructively proving certain existentially quantified theorems. We shall report on this in a subsequent publication.

4.3 Correspondence Theory

Many non classical logics are characterized by a basic Hilbert calculus which corresponds to a model theoretic semantics in terms of possible worlds. Different variants of the logic manifest themselves by additional Hilbert axioms which correspond to particular properties of the possible worlds structure.

For example normal modal logic [Che80] is characterized by the Hilbert axioms and rules for predicate logic plus the K-axiom

$$(\Box P \wedge \Box(P \Rightarrow Q)) \Rightarrow \Box Q$$

and the necessitation rule

$$\frac{P}{\Box P}.$$

The semantics of this logic is Kripke's well known possible worlds semantics:

$$w \models \Box P \quad \text{iff} \quad \forall v\ \mathcal{R}(w, v) \Rightarrow v \models P$$

where $\mathcal{R}$ is the accessibility relation.

The correspondence problem in these logics is to find for a given additional Hilbert axiom an equivalent property of the accessibility relation [vB84]. For example the Hilbert axiom $\forall P\ \Box P \Rightarrow P$ corresponds to the reflexivity of the accessibility relation. Actually this is one of the instances of power structure constructions. The predicates can be identified with the set of worlds where they are true and the modal operators can be seen as functions operating on these sets of worlds. For mechanizing deduction in non classical logics it is very important to find these correspondences [Ohl91]. So far the method for finding the correspondences was

mostly by intuition and the verification required complex proofs [vB84].

SCAN is the first algorithm which offers a method for computing the correspondences fully automatically. Moreover, since SCAN preserves equivalences, the computed correspondence axioms are *guaranteed to be complete* in the sense that a formula is derivable in the Hilbert calculus if and only if it is valid in the frames which are models of the computed correspondence axiom. The idea is as follows: We take the model theoretic semantics for the operators and translate the Hilbert axioms into predicate logic (cf. [Ohl91]).

For example $\forall P \ \Box P \Rightarrow P$ is translated into $\forall P \ (\forall w, v \ \mathcal{R}(w,v) \Rightarrow P(v)) \Rightarrow P(w)$. Application of SCAN to the negation of this formula and negating the result again yields the expected reflexivity axiom $\forall w \ \mathcal{R}(w,w)$.

Notice that this method is applicable as long as the semantics of the operators can be formalized with first-order predicate logic axioms. Moreover since SCAN preserves equivalence we automatically get soundness *and completeness* if the basic semantics is complete. That means the Hilbert calculus and the semantics together with the correspondence axioms computed by SCAN define the same logic.

Let us illustrate this application of SCAN with some more examples from modal logic.

In the sequel $H(P,v)$ means P holds in world v. For atomic P it is possible to write $P(v)$ instead of $H(P,v)$. The semantics of the modal operators are written as an equivalence in terms of the H-predicate and the accessibility relation. This equivalence is used as a rewrite rule for eliminating the operators. As usual, the $\Diamond$-operator is seen as an abbreviation for $\neg\Box\neg$.

Example 4.3 (Modal K4-Axiom)

Hilbert axiom: $\forall P \ \Box P \Rightarrow \Box\Box P$
H-Formulation: $\forall P \ \forall a \ H(\Box P \Rightarrow \Box\Box P, a)$

Semantics:
$\forall P \ \forall w \ H(\Box P, w) \Leftrightarrow (\forall v \ \mathcal{R}(w,v) \Rightarrow H(P,v))$

negated axiom: $\exists P \ \exists a \ \neg H(\Box P \wedge \Diamond\Diamond\neg P, a)$

translated (i.e. Semantics applied as rewrite rule).
$\exists P \ \exists a \ (\forall v \ \mathcal{R}(a,v) \Rightarrow P(v)) \wedge$
$\exists b \ \mathcal{R}(a,b) \wedge \exists c \ \mathcal{R}(b,c) \wedge P(c)$

clause form: $\neg\mathcal{R}(a,v), P(v)$
$\mathcal{R}(a,b)$
$\mathcal{R}(b,c)$ (Quantifier prefix:
$\neg P(c)$ $\exists a, b, c \ \forall v$)

P resolved away: $\neg\mathcal{R}(a,c)$
$\mathcal{R}(a,b)$
$\mathcal{R}(b,c)$

unskolemized:
$\exists a, b, c \ \neg\mathcal{R}(a,c) \wedge \mathcal{R}(a,b) \wedge \mathcal{R}(b,c)$
negated:
$\forall a, b, c \ \mathcal{R}(a,b) \wedge \mathcal{R}(b,c) \Rightarrow \mathcal{R}(a,c)$ (transitivity)

◁

SCAN can also work with quantified versions of modal logic. The predicate $exists(w,x)$ expresses that the object x exists in the domain of the world w.

Example 4.4 (Barcan Formula)

Hilbert axiom: $\forall P \ \Box\forall x \ P(x) \Rightarrow \forall x \ \Box P(x)$
H-Formulation: $\forall P \ \forall a \ H(\Box\forall x \ P(x) \Rightarrow \forall x \ \Box P(x), a)$

semantics of $\forall$:
$w \models \forall x \ P(x)$ iff for all a in the domain of w
$w[x/a] \models P(x)$.

negated axiom: $\exists P \ \exists a \ \neg H(\Box\forall x \ P(x) \wedge \exists x \ \Diamond\neg P(x), a$

translated:
$\exists P \ \exists a \ \forall v \ \mathcal{R}(a,v) \Rightarrow (\forall x \ exists(v,x) \Rightarrow P(v,x)) \wedge$
$\exists x \ exists(a,x) \wedge \exists b \ \mathcal{R}(a,b) \wedge \neg P(b,x)$

clause form: $\neg\mathcal{R}(a,v), \neg exists(v,x), P(v,x)$
$exists(a,c)$
$\mathcal{R}(a,b)$
$\neg P(b,c)$

P resolved away: $\mathcal{R}(a,b)$
$exists(a,c)$
$\neg exists(b,c)$

Here we have resolved with $\mathcal{R}(a,b)$.
This is an equivalence preserving simplification (subsumption resolution.)

unskolemized:
$\exists a, b, c \ \mathcal{R}(a,b) \wedge \neg exists(b,c) \wedge exists(a,c)$

negated:
$\forall a, b \ \mathcal{R}(a,b) \Rightarrow (\forall c \ exists(a,c) \Rightarrow exists(b,c))$
(Increasing Domain)

◁

The next example is the McKinsey axiom. It does not correspond to a first-order definable property of the accessibility relation.

Example 4.5 (McKinsey Axiom)

Hilbert axiom: $\forall P \ \Box\Diamond P \Rightarrow \Diamond\Box P$
H-Formulation: $\forall P \ \forall a \ H(\Box\Diamond P \Rightarrow \Diamond\Box P, a)$

semantics:
$\forall P \ \forall w \ H(P,w) \Leftrightarrow (\forall v \ \mathcal{R}(w,v) \Rightarrow H(P,v))$

negated axiom: $\exists P \ \exists a \ H(\Box\Diamond P \wedge \Box\Diamond\neg P, a)$
translated:
$\exists a \ \forall x \ \mathcal{R}(a,x) \Rightarrow \exists y \ (\mathcal{R}(x,y) \wedge P(y)) \wedge$
$\forall x \ \mathcal{R}(a,x) \Rightarrow \exists y \ (\mathcal{R}(x,y) \wedge \neg P(y))$

clause form: $\neg\mathcal{R}(a,x), \mathcal{R}(x,f(x))$
$\neg\mathcal{R}(a,x), P(f(x))$

$\neg\mathcal{R}(a,y), \mathcal{R}(y,g(y))$
$\neg\mathcal{R}(a,y), \neg P(g(y))$
(Quantifier prefix: $\exists f, g\ \forall x, y$)

P resolved away: $\neg\mathcal{R}(a,x), \mathcal{R}(x,f(x))$
$\neg\mathcal{R}(a,y), \mathcal{R}(y,g(y))$
$\neg\mathcal{R}(a,x), \neg\mathcal{R}(a,y), f(x) \neq g(y)$

unskolemized with second–order Henkin quantifiers:

$$\exists a \left(\begin{array}{c} \exists f\ \forall x \\ \exists g\ \forall y \end{array} \right) \begin{array}{l} (\mathcal{R}(a,x) \Rightarrow \mathcal{R}(x,f(x))) \wedge \\ (\mathcal{R}(a,y) \Rightarrow \mathcal{R}(y,g(y))) \wedge \\ (\neg\mathcal{R}(a,x) \vee \neg\mathcal{R}(a,y) \vee f(x) \neq g(y)) \end{array}$$

negated:

$$\forall a \left(\begin{array}{c} \forall f\ \exists x \\ \forall g\ \exists y \end{array} \right) \begin{array}{l} ((\mathcal{R}(a,x) \Rightarrow \mathcal{R}(x,f(x))) \wedge \\ (\mathcal{R}(a,y) \Rightarrow \mathcal{R}(y,g(y)))) \Rightarrow \\ (\mathcal{R}(a,x) \wedge \mathcal{R}(a,y) \wedge f(x) = g(y)) \end{array}$$

◁

The reader may verify with SCAN that the axiom $\forall P\ \Diamond\Box P \Rightarrow \Box\Diamond P$ is equivalent to confluence of $\mathcal{R}$: $\forall a,b,c\ \mathcal{R}(a,b) \wedge \mathcal{R}(a,c) \Rightarrow \exists d\ \mathcal{R}(b,d) \wedge \mathcal{R}(c,d)$. Compared to the McKinsey axiom, slight changes in the sequence of the modal operators obviously may have dramatic effects.

Based on Ackermann's first elimination method Andrzej Szałas has developed an alternative algorithm for computing correspondence axioms in modal logic [Sza92]. This algorithm terminates and reports failure in cases where SCAN loops. This may be an advantage. It may also be a disadvantage because sometimes it is possible to recognize and exploit regular structures in the loop.

Inspired by the SCAN algorithm, Harold Simmons has recently proposed another method for computing correspondence axioms for modal logics [Sim92]. His method extends an idea of van Benthem where it is necessary to guess appropriate instantiations for universally quantified predicate variables [vB84]. Simmons figured out how to determine these instantiations without guessing.

4.4 Semantics for Hilbert Calculi

There are more and more applications of logic in areas other than mathematics and classical computer science. In particular AI approaches to modelling intelligent agents are a very interesting area for applications of logic. Existing logics, however, usually do not cover all aspects which are needed there (formal notions of knowledge, belief, actions, causality, probability, time etc. as well as all kinds of combinations). Therefore there is a need for supporting the development of new logics and the mechanization of these logics. In [GO92a] we shall present a methodology for developing mechanizations of logics defined via Hilbert calculi. The main problem here is to find a model theoretic semantics which serves as a basis for a translation of the formulae of the new logic into predicate logic. This can be done by starting with a very general and therefore very weak neighbourhood semantics and by proving certain key lemmata which enable the transition form a weak semantics to a stronger semantics. A version of neighbourhood semantics for an n-place connective C is:
$w \models C(P_1,\ldots,P_n,w)$ iff
$\mathcal{N}(w,\zeta(v_1,\ldots,v_n)\Psi_C(v_1 \models P_1,\ldots,v_n \models P_n)$
which means that the set of all n-tuples $v_1,\ldots,v_n$ for which $\Psi_C(v_1 \models P_1,\ldots,v_n \models P_n)$ holds forms a neighbourhood of w. Ψ is a propositional function such as $\Rightarrow$, $\wedge$ etc. (For a binary causality operator, for example, one would choose $\Rightarrow$.)

A relational semantics for n-place connectives is

$w \models C(P_1,\ldots,P_n,w)$ iff
$(\forall v_1,\ldots,v_n\ R(w,v_1,\ldots,v_n)$ implies
$\Psi_C(v_1 \models P_1,\ldots,v_n \models P_n)$

where R_C is an $n+1$–ary accessibility relation.

Semantics of these kind can be used to translate the formulae of the new logic into predicate logic. Applied to Hilbert axioms, second–order formulae are obtained which can be processed by SCAN. The key lemmas for strengthening the semantics are: Closedness under intersection and supersets allows the transition from neighbourhood semantics to relational semantics. If the accessibility relation collapses to a point relation, relational semantics can be strengthened to predicate logic semantics.

We illustrate the idea with an example from Lukasiewicz. The implicational fragment of propositional logic can be axiomatized by modus ponens: from P and $P \rightarrow Q$ derive Q, and one more axiom $((P \rightarrow Q) \rightarrow R) \rightarrow ((R \rightarrow P) \rightarrow (S \rightarrow P))$

Assume the strongest semantics of the $\rightarrow$–connective (material implication) is not yet known, but we have an approximation in terms of possible worlds and a ternary relation:

$x \models P \rightarrow Q$ iff
$\forall y,z\ \mathcal{R}(x,y,z)$ implies $y \models P$ implies $z \models Q$

This semantics can be used to translate the above Hilbert axiom and modus ponens into predicate logic. For example the translation of modus ponens yields

$$\forall P,Q\ \forall x\ (P(x) \wedge (\forall y,z\ \mathcal{R}(x,y,z) \Rightarrow P(y) \Rightarrow Q(z)) \Rightarrow Q(x)$$

If we do this for the above axiom also, we get two second–order formulae from which SCAN can eliminate the quantifiers. The result is:

SCAN(Modus Ponens) =
$\forall z\ \exists x,y\ \mathcal{R}(x,y,z)$
SCAN(Axiom) = $\forall a,b,c,d,e,h,k$
$(\mathcal{R}(a,b,c) \wedge \mathcal{R}(c,d,e) \wedge \mathcal{R}(e,h,k)) \Rightarrow$
$(\exists u,v\ (\mathcal{R}(b,u,v) \wedge \mathcal{R}(d,v,k) \wedge$
$(\forall x,y\ \mathcal{R}(u,x,y) \Rightarrow (k = x))))$

It can now be proved with standard predicate logic means[4] that the conjunction of these two formulae is equivalent to

$$\forall a\ \mathcal{R}(a,a,a) \wedge \forall a,b,c\ \mathcal{R}(a,b,c) \Rightarrow a = b \wedge a = c$$

That means the relation $\mathcal{R}$ collapses to a point relation. This fact is the key lemma from which it is trivial to show that the $\rightarrow$ connective is actually material implication.

That means using SCAN and standard predicate logic theorem proving, both of which can be automatized, it is possible to analyze unknown logics and to find the strongest semantics. This semantics (plus some further optimizations) in turn serve as a basis for a translation into predicate logic.

5 Limitations of the SCAN Algorithm

There is a strange phenomenon which requires further investigation. From modal logic we know cases where a Hilbert axiom has a semantic property which is only second–order axiomatizable. Together with a further axiom, both correspond to a first–order axiomatizable property of the accessibility relation. For example the McKinsey axiom (example 4.5) $\forall P\ \Box\Diamond P \Rightarrow \Diamond\Box P$ alone corresponds to a second–order property of the accessibility relation (reversing skolemization in the SCAN algorithm needs second–order Henkin quantifiers). Combined with the transitivity axiom $\Box P \Rightarrow \Box\Box P$ (ex. 4.3), however, these two define atomicity $\forall x\ \exists y\ (\mathcal{R}(x,y) \wedge \forall z\ \mathcal{R}(y,z) \Rightarrow z = y))$ [vB84, page203] which is obviously a first–order definable property.

Applied to the McKinsey axiom, SCAN actually computes this property if the critical clause which prevents reversing the skolemization in the normal way is replaced with its factor. Although we have some ideas, why transitivity might in this particular case enable this operation, we are far from having a general theory for processing combinations of axioms with these strange properties. Actually the proof that the McKinsey axiom together with the transitivity axiom correspond to atomicity requires the axiom of choice. Therefore no simple solution of this problem is to be expected.

The example, however, shows that SCAN is not complete in the sense that it computes always a first–order formula when there is one. It is, however, not clear whether this has to be the job of SCAN at all. Obviously the conjunction of two formulae, one or both of which are second–order can be equivalent to a first–order formula, even if one of them alone is not equivalent to a first–order formula. This has to be investigated further.

[4]Thanks to Mark Reynolds who helped finding the proof.

In cases where SCAN loops and produces infinitely many clauses it is sometimes possible to recognize some regularities. We can try to recognize these regularities automatically and to exploit them for finding a finite representation of the infinitely many clauses.

6 Conclusion

We have presented an algorithm for the elimination of second–order quantifiers over predicate variables. The algorithm is simple and easy to implement by a slight modification of a standard resolution theorem prover. In fact, for example the theorem prover OTTER can be used without any changes [McC90]. The trick is to encode those predicates which are not to be resolved upon as $ans–literals. For each 'empty' clause, OTTER finds, it prints the $ans–literals which are part of the empty clause. These literals make up one clause of the result. In order to simulate C-resolution by ordinary resolution the input literals have to be abstracted, i.e. in the clause form the literals $P(s)$ have to be replaced with $P(x)|x \neq s$.

Although this works, one would like to have some changes. For example subsumption must be disabled because the implemented subsumption algorithm does not consider the $ans–literals and therefore removes clauses which actually are not subsumed. Furthermore there are no simplifications on the $ans–literals. This may cause the output to be unnecessary long.

An even simpler way for realizing SCAN is to let a theorem prover exhaustively generate all resolvents and then to collect manually those clauses not containing the predicates to be eliminated. This permits the application of all simplifications. On the other hand, it sometimes does not prevent the theorem prover from looping in parts of the clause set which are not to be touched by the SCAN algorithm.

We have shown that the SCAN algorithm can be applied to really hard problems such as computation of interpolands or computation of first–order circumscription. It can also be used to transform Hilbert axioms for non classical logics into properties of the semantic structure. This is extremely useful for the mechanization of these logics because it allows for compiling formulae of these logics into first–order predicate logic and using standard predicate logic deduction systems.

Convinced that these are not the only applications of our new technique we would like to encourage everybody to join our work in this area.

Acknowledgment

This work was stimulated considerably by many discussions with Luis Fariñas del Cerro and Andreas Herzig from Toulouse. We are grateful to Andrzej Szałas from Warsaw university who has digged out Ackermann's paper.

References

[Ack35a] Wilhelm Ackermann. Untersuchung über das Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 110:390–413, 1935.

[Ack35b] Wilhelm Ackermann. Zum Eliminationsproblem der mathematischen Logik. *Mathematische Annalen*, 111:61–63, 1935.

[BGW92] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Theorem proving for hierarchic first-order theories, 1992. To appear in Proc. ALP'92, Lecture Notes in Comp. Science.

[Bri92] Chris Brink. *Power Structures and Their Applications*. PhD thesis, Fac. of Science, Rand Afrikaans University, Johannesburg, South Afrika, 1992.

[Che80] B. F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, Cambridge, 1980.

[GO92a] Dov M. Gabbay and Hans Jürgen Ohlbach. From a Hilbert calculus to its model theoretic semantics, 1992. presented at UKALP, april 92, to be published.

[GO92b] Dov M. Gabbay and Hans Jürgen Ohlbach. Quantifier elimination in second–order predicate logic. *South African Computer Journal*, 7:35–43, July 1992.

[Hen61] L. Henkin. Some remarks on infinitely long formulas. In *Infinistic Methods*, pages 167–183. Pergamon Press, Oxford, 1961.

[KK66] G. Kreisel and J.L. Krivine. *Éléments de Logique Mathématique. Théorie des modèles*. Société Mathématique de France, 1966.

[Lif85] Vladimir Lifschitz. Computing circumscription. In *Proc. of IJCAI 85*, pages 121–127. University of, 1985.

[McC80] John McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:295–323, 1980.

[McC90] William McCune. OTTER 2.0. In Mark Stickel, editor, *Proc. of* 10^{th} *Internation Conference on Automated Deduction, LNAI 449*, pages 663–664. Springer Verlag, 1990.

[Ohl91] Hans Jürgen Ohlbach. Semantics based translation methods for modal logics. *Journal of Logic and Computation*, 1(5):691–746, 1991.

[OS91] Hans Jürgen Ohlbach and Jörg H. Siekmann. The Markgraf Karl refutation procedure. In Jean Luis Lassez and Gordon Plotkin, editors, *Computational Logic, Essays in Honor of Alan Robinson*, pages 41–112. MIT Press, 1991.

[Sim92] Harold Simmons. An algorithm for eliminating predicate variables from π^1_1 sentences, 1992.

[Sza92] Andrzej Szałas. On correspondence between modal and classical logic: Automated approach. Technical Report MPI–I–92–209, Max Planck Institut für Informatik, Saarbrücken, march 1992.

[vB84] Johan van Benthem. Correspondence Theory in D. Gabbay, F. Guenthner: Handbook of Philosophical Logic, volume II, Extensions of Classical Logic of *Synthese Library Vo. 165*, pages 167–248. D. Reidel Publishing Company, Dordrecht, 1984.

VII.

Logics of Belief and Intention

An Abstract Architecture for Rational Agents

Anand S. Rao
Australian Artificial Intelligence Institute
1 Grattan Street
Carlton 3053, Australia
anand@aaii.oz.au

Michael P. Georgeff
Australian Artificial Intelligence Institute
1 Grattan Street
Carlton 3053, Australia
georgeff@aaii.oz.au

Abstract

A number of architectures have been proposed for systems that are situated in dynamic environments and have to reason, plan, and act under stringent time constraints. In this paper, we consider architectures based on the notion of a rational agent, in which the system is viewed as having attitudes of belief, desire, and intention, which in turn influence its decision making and determine its behavior. A basic interpreter for situated systems is first described. This interpreter is then specialized to form an abstract belief-desire-intention (BDI) architecture, and it is shown how this interpreter satisfies some of the theoretical properties of proposed BDI formalisms. The abstract interpreter is developed further to produce different types of rational agents, based on variations in their commitment strategies. Finally, the design of a practical system is considered, taking into account the complexity limits on an agent's deductive and planning capabilities.

1 INTRODUCTION

Researchers in Artificial Intelligence have become increasingly interested in the design of systems that are situated or embedded in a changing environment. The major difficulty facing the designer of such systems is that any reasoning or decision making regarding the selection and execution of tasks is resource bounded. In particular, the necessity to complete tasks in a timely way requires an appropriate balance between time spent deliberating and time spent acting. Too long spent deliberating may seriously affect the ability of the system to complete its tasks. Moreover, the basis of the system's decisions is itself subject to change, providing further motivation for timely deliberation.

Various architectures for such systems have been proposed. Some of these attempt to avoid the problem of real-time deliberation by a "compilation" process that aims to make all necessary decisions about task selection and execution prior to operation of the system [2, 17]. However, there is no evidence to suggest that these systems can be successfully applied to problems involving complex task management. Further, they suffer the disadvantage of requiring recompilation (possibly to be performed under stringent time constraints) if the information upon which the compilation was based proves to be innaccurate.

The architectures that, instead, are specifically designed to allow real-time deliberation are numerous. A good review of these systems, though somewhat dated, is provided by Laffey et. al. [13]. Most of these approaches are based on a blackboard architecture, and many have the decision procedures "hard-wired" into the system.

An alternative approach, in which these decision procedures are more transparent and general, is based on the notion of a rational agent. In this approach, the system is viewed as having certain *mental attitudes* which in turn influence its decision making and determine its behavior. The simplest of these models, called BDI architectures, are based on attitudes of *belief*, *desire* and *intention*. The first two attitudes represent, respectively, the information state and the evaluative state of the agent. The last represents decisions the agent has made at a previous time, and is critical for achieving adequate or optimal performance when deliberation is subject to resource bounds [1, 12].

Recently, a number of attempts have been made to formalize these notions, to describe how these mental attitudes change in response to the agent's changing environment, and to show how they determine the actions of the agent [3, 16].

Various computational systems based on these principles—at least at an intuitive level—have also been proposed [5, 7] and applied to a number of important practical problems, such as air traffic control [11], spacecraft systems handling [6], and telecommunications management [15]. However, these computational

systems are very complex, and there is little in the literature that attempts to relate these systems to the formalized models of BDI architectures.

This paper is an attempt to bridge this gap, and grew out of discussions during the last KR&R conference where concern was expressed regarding the decreasing number of papers that make a principled connection between theoretical ideas and implemented systems.

In Section 2, we first present a basic interpreter for situated systems. We then specialize the interpreter to form an abstract BDI architecture, and show how it satisfies some of the theoretical properties of BDI formalisms. In Section 3, we develop the abstract interpreter further to produce different types of rational agents, based on variations in their commitment strategies. Section 4 considers the design of a practical system, taking into account the complexity limits on deductive and planning capabilities. We conclude with a brief comparison to other work in this area.

2 ABSTRACT ARCHITECTURE

2.1 A BASIC INTERPRETER

In this section, we describe a basic abstract interpreter for situated systems.

The inputs to the system are *events*, and these are received via an *event queue*. The events the system can recognize (i.e., that can appear on the event queue) include both external events (events occurring in the environment) and internal events (events occurring within the system). External events may directly generate particular internal events, such as updating some component of the system state. We assume that the events are atomic in some sense, and that they are recognized after they have occurred (rather than, for example, during their occurrence).

The outputs of the system are *actions*, which are performed by an *execute* function. Without loss of generality, we assume actions are also atomic. Actions can be executed either successfully or unsuccessfully, and the events corresponding to both these behaviors may be recognized by the system (i.e., may appear in the event queue). However, in the basic interpreter, there is no requirement that either of these classes of events be recognized at all.

The system selects and executes *options*. Options correspond to programs or subroutines of a conventional programming system. In the general case, they may be tasks, plans of action, production rules, finite automata, or circuit networks.

Options are generated on the basis of the current system state and the events appearing in the event queue. For example, in a conventional programming system, the option-invoking events would be subroutine calls; in a forward-chaining production system, they would be the assertion of possible antecedents of the production rules.

The abstract interpreter is given below. We assume the procedures and functions appearing in the interpreter operate on the system state, denoted by S.

```
basic-interpreter
initialize-state();
do
    options := option-generator(event-queue, S);
    selected-options := deliberate(options, S);
    update-state(selected-options, S);
    execute(S);
    event-queue := get-new-events();
until quit.
```

The interpreter works by continuously performing the following cycle. First, the interpreter determines what options are available to be performed or acted upon. The next step of the interpreter is to decide which options to adopt (i.e., to commit to performing). This is achieved by the `deliberate` procedure.

Once the options have been selected, the system state is updated appropriately and some set of atomic actions executed. Exactly which atomic actions are executed will depend on the system state and the definition of the execute function. Thus, while the selected options determine the transformation of the system state, and thereby determine which actions get executed, not all the generated actions may appear as options.

Finally, the event queue is updated to contain all those recognisable events that have occurred during the cycle.

It is important to note that events are recognized (and thus able to be acted upon) only once per cycle. This means that the system's reaction time is bounded, from below, by the time taken to perform a cycle.

This abstract interpreter can be used as a basis for most situated systems, including those in which most of the deliberation among tasks has been performed at compile time [2, 17]. We discuss these various systems elsewhere [8]. Here, we specialize the interpreter to a BDI architecture.

2.2 AN ABSTRACT BDI INTERPRETER

In a BDI interpreter, the system state comprises three dynamic data structures representing the agent's beliefs, desires, and intentions. For simplicity, we assume that the agent's desires are mutually consistent, although not necessarily all achievable. Such mutually consistent desires will be called *intrinsic goals* or simply *goals*.

We take each of these to be abstract data struc-

tures allowing, in particular, update and query operations. The update operations in which we are interested include `belief-add`, `belief-remove`, `goal-add`, `goal-remove`, `intention-add`, and `intention-remove`. These operations should be self explanatory but will, in any case, become clear later. The update operations on beliefs, goals, and intentions are subject to respective compatibility requirements, represented by the functions `belief-compatible`, `goal-compatible`, and `intention-compatible`. These functions are critical in enforcing the formalized constraints upon the agent's mental attitudes.

In addition to refining the notion of system state, we also refine and extend the interpreter procedures. All these refinements act to maintain certain constraints on the dynamics of beliefs, goals, and intentions.

The main interpreter loop is given below. We assume that the belief, goal, and intention structures are global. They are represented by `B`, `G`, and `I`, respectively. We also assume that the event queue is global, so that the procedures occurring in the interpreter can directly update it. The procedure `get-new-external-events` now simply collects the external events (rather than both the external and internal events) that have occurred since its last call.

BDI-interpreter
initialize-state();
do
    options := option-generator(event-queue,B,G,I);
    selected-options := deliberate(options,B,G,I);
    update-intentions(selected-options,I);
    execute(I);
    get-new-external-events();
    drop-successful-attitudes(B,G,I);
    drop-impossible-attitudes(B,G,I);
until quit.

At the beginning of every cycle the option generator reads the event queue and returns a list of options. These need not be *all* the options available to the agent as determined by the agent's capabilities; rather, they are the ones chosen for further deliberation and possible execution. Next, the deliberator selects a subset of options to be adopted and adds these to the intention structure. If there is an intention to perform an atomic action at this point in time, the agent then executes it.

Any external events that have occurred during the interpreter cycle are then added to the event queue. Internal events are added as they occur.

Next, the agent modifies the intention and goal structures by dropping all successful goals and satisfied intentions, as well as impossible goals and unrealisable intentions.

The main difference between this interpreter and the basic interpreter is the presence of the last three procedures. These procedures eliminate a number of options that were carried over from one cycle to the next in the basic interpreter.

Before we can examine the logical properties exhibited by the above interpreter, we review the representation language we use to describe the mental attitudes of the system.

2.3 REPRESENTATION LANGUAGE

The representation language for the architecture [16] is an adaptation of Computation Tree Logic, CTL* [4]. This language is used as a meta-language for describing the mental state of the system and its behavior. The actual internal representation of beliefs, goals, and intentions is not of concern. The only requirement is that the internal representation have sufficient expressive power to enforce the constraints mentioned below.

We assume the existence of multiple worlds, each one of which is a temporal structure with a branching time future and a single past called a *time tree*. A particular time point in a particular world is called a *situation* [16].

A distinction is made between *state formulas* and *path formulas*: the former are evaluated at a specified time point in a time tree and the latter over a specified path in a time tree. State formulas can be either propositional formulas or first-order formulas. We introduce two modal operators, optional and inevitable, which operate on path formulas. A path formula ψ is said to be optional if, at a particular time point in a time tree, ψ is true of at least one path emanating from that point; it is inevitable if ψ is true of all paths emanating from that point.[1] The standard temporal operators $\bigcirc$ (next), $\Diamond$ (eventually), $\Box$ (always), and U (until) operate over state and path formulas.

Action types transform one time point of the temporal structure into another. Primitive (atomic) actions are those actions directly performable by the agent and uniquely determine the next time point in a time tree. Non-primitive actions map to non-adjacent time points, thus allowing us to model the partial nature of plans. Their potential for decomposition into primitive actions can be used to model hierarchical plan development, which will be investigated further in Section 4.

Agents may attempt to execute some action, but fail to do so successfully. We thus need to distinguish between the successful execution of actions and their failure. This is achieved by introducing the state formulae *succeeded*(a) and *failed*(a) to indicate a past successful attempt and a past failed attempt, respectively, to perform action a. The formula *done*(a) represents a

[1] In CTL*, E and A are used to denote these operators.

past successful or failed attempt to perform a and the formula $does(a)$ represents a future attempt to perform a.

Note that only the agent's actions label the transitions from one time point to another. External events do not appear explicitly as transitions. However, their occurrence can be represented as necessary by introducing a state predicate, $occurred(e)$, which is true of a state if and only if event e has just occurred.

All the above propositions can be objects of an agent's beliefs, goals, and intentions. The fact that an agent has a belief, goal, or intention towards a proposition ϕ is denoted BEL(ϕ), GOAL(ϕ), and INTEND(ϕ), respectively.

Each of these attitudes is modelled semantically by introducing a corresponding accessibility relation between possible worlds. An agent is said to believe a proposition ϕ (or have a goal or intention towards ϕ) just in case ϕ holds in all belief-accessible (respectively, goal-accessible or intention-accessible) worlds. We assume that the set of beliefs, goals, and intentions are separately closed with respect to the appropriate logic (e.g., if the formulas are first-order formulas, they are closed under first-order logic).

Note that the branching time structure is critical if we wish to distinguish between the choice of action available to the agent (represented by the branching time structure) and the uncertainty of an action's applicability or outcome (represented by multiple belief-accessible worlds). In contrast, the possible occurrences of different external events can be represented simply by different belief-accessible worlds, as the agent has no direct choice over their occurrence.

2.4 BASIC AXIOMS

We have given elsewhere [16] a semantic structure and an axiomatization for beliefs, goals, and intentions. In this section, we review some of these axioms and show how they are reflected in the abstract architecture. The most important of these axioms are given below:

(AI1) GOAL(α) $\supset$ BEL(α).
(AI2) INTEND(α) $\supset$ GOAL(α).
(AI3) INTEND($does(a)$) $\supset$ $does(a)$.
(AI4) INTEND(ϕ) $\supset$ BEL(INTEND(ϕ)).
(AI5) GOAL(ϕ) $\supset$ BEL(GOAL(ϕ)).
(AI6) INTEND(ϕ) $\supset$ GOAL(INTEND(ϕ)).
(AI7) $done(a)$ $\supset$ BEL($done(a)$).
(AI8) INTEND(ϕ) $\supset$ inevitable$\Diamond$($\neg$INTEND(ϕ)).

Axiom AI1 requires a restricted form of belief-goal compatibility regarding the options available to the agent. The formula α is so-called O-formula [16], which contains no positive occurrences of inevitable (or negative occurrences of optional) outside the scope of the modal operators BEL, GOAL, or INTEND. In other words, if the agent has a goal to optionally achieve a certain condition, the agent must also believe that this is an option.

This axiom is realized computationally in the abstract machine by the function `goal-compatible`. That is, Axiom AI1 serves as an abstract specification for this function. Correctly implemented, this function will only allow a new goal to be adopted if the agent believes it to be an option.

Axiom AI2 enforces a restricted form of goal-intention compatibility. Similar to the belief-goal compatibility axiom, Axiom AI2 serves as an abstract specification for the function `intention-compatible`, which only allows a new intention to be formed if it is a goal.

Axiom AI3 captures volitional commitment [1] by stating that the agent will act if it has an intention towards a single primitive action a. This is captured in the abstract architecture by the `execute` function. Note that we have not said that the action a will occur successfully, just that the agent is committed to *trying* it. Whether the agent is successful or not depends on the environment in which it is embedded.

Axioms AI4 and AI5 require an agent to be aware of its goals and intentions. These axioms are partial specifications for the abstract data structure representing the agent's beliefs. They are simply implemented by allowing the function representing the belief predicate to also return the goals and intentions of the agent. Axiom AI6 acts in a similar way as a partial specification for the data structure representing the agent's goals.

Axiom AI7 specifies that the `execute` function must post a description of $done(a)$ on the event queue. Alternatively, or in addition, we could adopt a stronger version of AI7, requiring an agent to know whether or not it has been successful in performing the chosen action. This can be realized by requiring the function `get-external-events` to post the success or failure of the action on the event queue.

Other restrictions are placed on the interpreter functions. In particular, the abstract data structures representing the agent's beliefs, goals, and intentions are required to conform to some update conditions that ensure they remain consistent.

We denote options by path formula, and require that the options suggested by the option generator be goals of the agent. That is, if ϕ is an option, then GOAL(optional ϕ) must hold. In addition, we require that the function `deliberate` return some subset of these. Given Axiom AI2, these restrictions prevent the generation of options that would subsequently be found to be incompatible with the systems goals.

The functions **drop-successful-attitudes** and **drop-impossible-attitudes** update the goal and intention structures on the basis of newly acquired beliefs and goals. These must be constructed to ensure that Axioms AI1 and AI2 are maintained as the system moves from one BGI state to the next. In addition, to capture Axiom AI8, **drop-impossible-attitudes** must eventually drop any intention that remains unsuccessful.

Provided each function appearing in the abstract interpreter satisfies the specifications given above, it is not difficult to prove that any behavior of the system will satisfy Axioms AI1 to AI8.

2.5 EXAMPLE

In this section, we present a simple example to illustrate the operation of the abstract interpreter. We assume an abstract machine whose beliefs, goals, and intentions are represented by formulas in propositional CTL*. Non-atomic actions are represented by propositions or sequences of actions. A proposition occurring in a sequence denotes any action (or sequence of actions) whose successful occurrence always yields a world state in which the proposition is true.

In this example, John acquires a goal to quench his thirst. He believes that he can satisfy this goal in one of two ways: (a) open the tap, fill the glass, and then drink water from the glass; or (b) get to a state where he has soda, fill the glass, and then drink soda from the glass. In the first case, his means of quenching thirst involves executing three primitive actions one after the other; in the second, he has to solve a subgoal (namely, to have soda) and then execute two primitive actions. To get to a state where he has soda, John has to open the refrigerator and remove the soda bottle.

John thus initially has the following beliefs regarding the effects of these actions:[2]

BEL(inevitable□(**have-soda;fill-glass;drink**)
⊃ **quenched-thirst**)
BEL(inevitable□(**open-tap;fill-glass;drink**)
⊃ **quenched-thirst**)
BEL(inevitable□(**open-fridge;remove-soda**)
⊃ **have-soda**)

He also believes that he has the option to carry out all these action sequences successfully:

BEL(optional◇(**have-soda;fill-glass;drink**))
BEL(optional◇(**open-tap;fill-glass;drink**))
BEL(optional◇(**open-fridge;remove-soda**))

Finally, he believes that the action of removing soda from the refrigerator cannot possibly succeed if the soda is not in the refrigerator:

BEL(inevitable□(¬(**soda-in-fridge**)
⊃ inevitable¬◇(**remove-soda**))).

(Of course, as a consequence of these beliefs, he must currently believe that the soda is in the refrigerator.)

In addition, John has the goal of quenching his thirst:

GOAL(inevitable◇(**quenched-thirst**)).

(For simplicity, we will assume that John's goals are closed with respect to the above beliefs.)

With the above state as the starting condition, we go through the interpreter cycle once. Assume that the goal to quench thirst has just been added as an intrinsic goal. The event queue will therefore include the event **goal-add**(inevitable◇(**quenched-thirst**)). Based on this event and the beliefs of the agent, the option generator will suggest two different options; namely, drink water and drink soda. The deliberator has to choose between these two options. Let us assume that it chooses the option to drink soda, and therefore adds this to the intention structure. Now John has the intention to quench his thirst by drinking soda. As a result, he also has the intention to obtain soda and thus to open the refrigerator.

The next step of execution results in the primitive action of opening the refrigerator. Let us assume that this action succeeds, but that John then discovers that the soda is not there. As this implies that the action **remove-soda** can never be successful, John will be forced to drop his current intentions.

In the next cycle, John adopts the only remaining option to drink water. He then proceeds as in the previous cycle.

In Figure 1, we give the state transitions of the abstract interpreter for this example. The state is registered at the beginning of each interpreter cycle. We only record external events, all of which are successful. The modal operators are omitted when it is clear from the context, and only changes to the structures are indicated. $\mathcal{B}$ represents the initial belief state as described above.

3 MODELLING RATIONAL AGENTS

So far, while we have discussed various constraints among an agent's beliefs, goals, and intentions, we have not required the agent to be committed to its intentions. Thus, in the example given above, John could have adopted the option of drinking water after the refrigerator was opened, even had the soda been present. Instead, we would like to consider agents who commit to their intentions, thereby reducing the options that need be deliberated upon.

[2] As all the properties stated here and throughout the paper refer to successful actions, we will omit the predicate *succeeded* when it is clear from the context.

BEL	GOAL	INTEND	*done*	*succeeded*
B	◇(quenched-thirst)	–	–	–
¬ ◇(remove-soda)	unchanged	–	open-fridge	open-fridge
unchanged	unchanged	fill-glass; drink	open-tap	open-tap
unchanged	unchanged	drink	fill-glass	fill-glass
quenched-thirst	–	–	drink	drink

Figure 1: State Trace for the Abstract BDI Interpreter

In this section, we review some of the commitment axioms presented in previous work [16] and show how they can be realized by the abstract interpreter.

3.1 BLIND COMMITMENT

We defined a *blindly* committed agent to be one that maintains its intentions until it *actually* believes that it has achieved them. Formally, the axiom of blind commitment states that, if an agent intends that inevitably ϕ be eventually true, then the agent will inevitably maintain its intentions until it believes ϕ.

(AI9a) $\mathsf{INTEND}(\mathsf{inevitable}\Diamond(\phi)) \supset$
$\mathsf{inevitable}(\mathsf{INTEND}(\mathsf{inevitable}\Diamond(\phi))\ \mathsf{U}\ \mathsf{BEL}(\phi))$.

To realize the above axiom, the blindly committed agent must: (a) only consider options that are consistent with its current intentions; (b) not entertain any changes to its beliefs that conflict with its current intentions; and (c) not entertain any changes to its goals that conflict with its current intentions.

We accomplish this by adding further requirements to the abstract specifications for the interpreter functions given in Section 2.4. First, we require that the option generator only return options that are consistent with the agent's current intentions.

In addition, the belief and goal compatibility functions are further specialized so that new beliefs and goals can only be added or removed if they, too, are consistent with the agent's current intentions. More formally, the function `belief-compatible` returns false for the event `belief-add`(ϕ) if, for any O-formula α such that $\mathsf{INTEND}(\alpha)$, it is provable that the addition of $\mathsf{BEL}(\phi)$ yields $\mathsf{BEL}(\neg\alpha)$. Similar requirements hold for the removal of beliefs and for the addition and removal of goals.

The function `drop-impossible-attitudes` must also be modified to ensure that, when dropping an intention towards a proposition ϕ, the agent believes that ϕ has been achieved (if necessary, by revising the agent's current set of beliefs to accomodate a belief in ϕ).

Consider the previous example, at the point where John is about to fill his glass with water. Assume that John normally adopts as goals any requests made by Mary, and that Mary currently requests that he not fill the glass. The attempt to add this new goal (of not filling his glass) will be checked for compatibility. As this conflicts with John's current intention to drink the water, the `goal-add` event will be considered incompatible and the goal will not be added to the goal structure.

Similarly, consider the previous example where John sees that there is no soda in the refrigerator. Despite what he sees, John will be unable to adopt this new belief as it conflicts with his current intentions. He will thus pursue his chosen course of action, eventually coming to believe he has successfully quenched his thirst even though he may not have.

3.2 SINGLE-MINDED COMMITMENT

A blind-commitment strategy is clearly very strong: the agent will eventually come to believe it has achieved its intentions no matter with what information it is supplied. Relaxing this requirement, one can define *single-minded* commitment, in which an agent maintains its intentions as long as it believes that they are still options. More formally:

(AI9b) $\mathsf{INTEND}(\mathsf{inevitable}\Diamond(\phi)) \supset$
$\mathsf{inevitable}(\mathsf{INTEND}(\mathsf{inevitable}\Diamond(\phi))\ \mathsf{U}$
$(\mathsf{BEL}(\phi) \vee \neg\mathsf{BEL}(\mathsf{optional}\Diamond(\phi))))$.

We construct such an agent by modifying the function `belief-compatible` to always return true and allowing `drop-impossible-attitudes` to drop intentions that are no longer believed to be options (this latter situation could never arise for the blindly-committed agent). All the other functions are as above.

Consider the previous example of John and Mary. If John is a single-minded agent, he will entertain changes in beliefs that conflict with his current intentions, and hence come to believe that there is no soda in the refrigerator. As a result he will drop his intention to remove the soda from the refrigerator. However, John does not entertain changes in goals that conflict with his current intentions. Hence, he will not drop the intention to fill his glass with water solely upon Mary's request to do so.

3.3 OPEN-MINDED COMMITMENT

A single-minded agent will not drop its intentions as long as it believes its intentions are still achievable. In

other words, a single-minded agent is committed to its goals (that is, does not allow incompatible goals to be added or removed) but is open to changes in its beliefs.

We define an *open-minded* agent to be one that is prepared to change its goals as well as its beliefs, but otherwise maintains its intentions as long as these intentions are still part of its goals. The axiom of open-minded commitment can be stated as follows:

(AI9c) INTEND(inevitable$\Diamond(\phi)$) $\supset$
inevitable(INTEND(inevitable$\Diamond(\phi)$) U
(BEL(ϕ) $\vee$ $\neg$GOAL(optional$\Diamond(\phi)$))).

Such an agent can be constructed by modifying the function `goal-compatible` to return true except where the addition or removal of the goal would violate Axiom AI1 (Section 2.4). The function `drop-impossible-attitudes` will now drop intentions if they are not either belief or goal options.

In the example, if John is an open-minded agent, he will entertain changes to beliefs and goals that conflict with his current intentions. As a result, on adopting the goal requested by Mary, he will drop his intention to fill the glass with water and consider some other option instead.

4 A PRACTICAL SYSTEM ARCHITECTURE

The abstract architecture presented in the last two sections is a useful abstraction of the theoretical model presented by us elsewhere [16]. It shows the interrelationships among the beliefs, goals, and intentions of an agent. More importantly, it illustrates various components of practical reasoning [1]; namely, option generation, deliberation, execution, and intention handling.

However, it is not a practical system for rational reasoning. The architecture is based on a (logically) closed set of beliefs, goals, and intentions and the provability procedures required are not computable. Moreover, we have given no indication of how the option generator and deliberation procedures can be made sufficiently fast to satisfy the real-time demands placed upon the system.

In this section, we make a number of important choices of representation which, while constraining expressive power, provide a more practical system for rational reasoning. The system presented is a simplified version of the Procedural Reasoning System (PRS) [7, 10].

4.1 BELIEFS AND GOALS

The system operates only on *explicit* beliefs and goals and not on their closure. Further, we identify a subset of the agent's beliefs and goals, which we call *current*. These are taken to be a ground set of literals (i.e., ground predicate formulas and their negations) with no disjunctions or implications. Intuitively, they represent beliefs and goals that are currently held, but which can be expected to change over time.

It may seem that such a language is too simple to be of much practical use. As we will see shortly, however, implications and variables can be introduced elsewhere with little loss of expressiveness, but for a substantial gain in control.

4.2 PLANS

In the previous abstract interpreter, information about the means of achieving certain future world states and the options available to the agent were represented as beliefs of the agent. We now represent these forms of belief as *plans*. Intuitively, plans are abstract specifications of both the means for achieving certain goals and the options available to the agent.

Each plan has a name, its *plan type*. The method of carrying out the plan is called the *body* of the plan and is specified by a plan graph. A *plan graph* is a rooted, directed, acyclic graph in which each edge is labeled with a simple plan expression. A simple plan expression is either a primitive action or a proposition describing a world state that is to be achieved by execution of some other plan.

The conditions under which a plan can be chosen as an option are specified by an *invocation condition* and a *precondition* . The invocation condition specifies the "triggering" event that is necessary for invocation of the plan, and the precondition specifies the situation that must hold for the plan to be executable. The successful execution of a plan results in a set of atomic formulas being believed and a set of atomic formulas being no longer believed, specified respectively by the *add list* and the *delete list* of the plan.

As described, plans represent not only beliefs about what sequences of actions achieve certain conditions, but also what options are available to the agent. That is, a plan represents, firstly, the belief that, if the invocation condition and precondition are satisfied and each action in the body of the plan is successfully executed, then the propositions appearing in the add list will be true at completion of execution. Secondly, having a plan also means that the body of the plan is believed to be an option whenever the invocation condition and precondition of the plan are satisfied. Plans thus represent a number of beliefs that we previously described using complex modal formulae.

Implications can also be represented as plans, with the premise being the precondition of the plan and the consequence being the add list. As preconditions are nothing but complex conditions on the agent's beliefs, the agent can execute plans to compute new conse-

quences. These consequences can further trigger other plans and hence be used to infer further consequences. In this manner, the agent can close its set of beliefs. This gives the agent greater control as to when to compute consequences of its current beliefs and to what extent to close its set of beliefs, goals, and intentions.

The plans for the running example (slightly simplified) are shown in Figure 2.

4.3 INTENTIONS

Plans provide a hierarchical structure and allow tractable real-time option generation and means-end reasoning. The options produced by the option generator, and selected by the deliberator, are plans. As they are adopted, they are added to the intention structure. Thus, intentions are represented as sets of hierarchically related plans.

More particularly, to achieve an intended end (adopted goal), the agent forms an intention towards the means of achieving the end; namely, the plan body of an appropriate plan. This means-end pair, together with information about variable bindings and control points, is called an *intention frame*.

An intention towards a means results in the agent adopting another end (subgoal) and the means for achieving this end, thus creating another intention frame. This process continues until the subgoal can be directly executed, i.e., until we reach a primitive action. The next subgoal in the plan is then attempted.

Just as in conventional programming languages containing subroutines, a stack of such frames is used to keep track of variable bindings and control points. We call this an *intention stack*. Each intention stack thus represents a separate process or task. These intention stacks are organized into an *intention structure*, which places various ordering constraints on the intention stacks.

It is important to note that intention stacks are created not just by the adoption of intrinsic goals; any event that appears in the invocation condition of a plan can trigger the development of an intention stack. This capability is important if the system is going to be responsive to external events without mediating everything through the establishment of intrinsic goals.

4.4 A PRACTICAL INTERPRETER

The main interpreter loop for this system is identical to the one discussed previously. However, as the system is embedded in a dynamic environment, the procedures appearing in the interpreter must be fast enough to satisfy the real-time demands placed upon the system. We show below how the choice of representation allows this to be achieved for the critical procedures of option generation and deliberation.

Given a set of trigger events from the event queue, the option generator iterates through the plan library of the agent and returns those plans whose invocation condition matches the trigger event and whose preconditions are believed by the agent. Note that the provability procedure used here involves simple unification with the beliefs of the agent.

```
option-generator(trigger-events)
options := {};
for trigger-event ∈ trigger-events do
  for plan ∈ plan-library do
    if matches(invocation(plan),trigger-event) then
      if provable(precondition(plan),B) then
        options := options ∪ {plan};
return(options).
```

The deliberate procedure can be based on any decision algorithm that has an execution time whose bounds conform with the time constraints of the environment. It can be made sufficiently fast simply by using increasingly less complex deliberation methods. In the extreme, random choice can be used.

Clearly, however, it is sometimes necessary to carry out lengthy deliberation to select between options. This kind of deliberation can be achieved by including appropriate deliberation procedures (called metalevel plans) in the plan library of the agent. Thus the `deliberate` procedure may select, and thus form an intention towards, various metalevel plans for performing more complex deliberation than it itself is capable. Such a deliberation procedure is adopted by PRS and is discussed in more detail elsewhere [9]. We give a simplified version below:

```
deliberate(options)
if length(options) ≤ 1 then return(options);
else metalevel-options :=
     option-generator(belief-add(option-set(options)));
  selected-options := deliberate(metalevel-options);
  if null(selected-options) then
    return(random-choice(options));
else return(selected-options).
```

Note that there can be more than one metalevel option which results in the procedure being called recursively until at most one option remains. In the case that there are no metalevel options available, the deliberator simply makes a random choice.

The process of option generation can be simplified by inserting an additional procedure, `post-intention-status`, at the end of the interpreter loop. The purpose of this procedure is to delay posting events on the event queue regarding any changes to the intention structure until the end of the interpreter loop. The reason for so doing is that, in a single cycle, the intention structure can undergo a large number of changes, many of which become irrelevant in the light

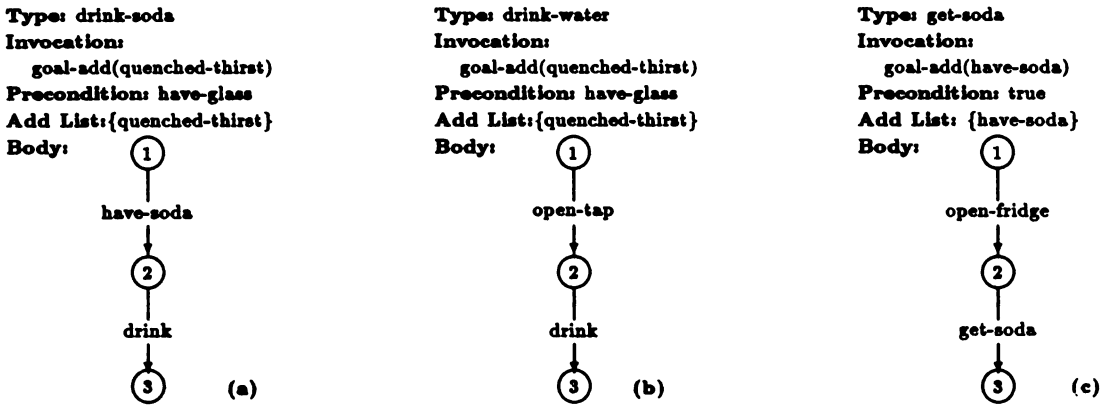


Figure 2: Plans for quenching thirst.

of later changes. By waiting until the end of the cycle, only the final changes to the intention structure need be notified.

In the abstract interpreter, commitment is achieved by reducing the options generated by the option generator. As events serve the basis upon which options are generated, the events that appear in the event-queue have a critical impact on option generation.

In particular, by posting appropriate events to the event queue, `post-intention-status` can determine, among other things, which elements of the intention structure will be noticed by the option generator. In this way, the choice of the procedure `post-intention-status` yields various notions of commitment and results in different behaviors of the agent.

A simple form of `post-intention-status` is given below.

```
post-intention-status()
if null(I) then
  for goal ∈ G do
    event-queue := event-queue ∪ goal-add(goal);
else for stack ∈ I do
    event-queue :=
      event-queue ∪ goal-add(means(top(stack))).
```

It is important to note that this interpreter behaves differently to the interpreter described in Section 3 in a crucial way. In the above interpreter, any attempt to close the beliefs, goals, or intentions of the agent must take place over potentially an arbitrary number of cycles. In contrast, such closure took place in a single cycle in the previous interpreter. Thus, for the example given in Section 2.5, it would take several cycles of the above interpreter before the primitive action of opening the refrigerator door was attempted, whereas previously this was done on the first cycle.

Assuming that external events can occur within the cycle interval, this means that the above interpreter will not strictly obey the axioms given in Section 2.4. But this is exactly what one really wants: if one wishes to guarantee responsiveness, one must be able to interrupt deliberation processes that are potentially time unbounded. In the limit, where provabilty is determined so fast that no external events can occur during the process, the axiomatization is correctly realized.

Consider the previous example with plans as shown in Figure 2. Assume that the goal `quenched-thirst` has just been added as an intrinsic goal. The event queue will contain the event `goal-add(quenched-thirst)`. As the invocation conditions of `drink-soda` and `drink-water` match with the trigger event and their context conditions are believed by the agent, the option generator returns both these plans as suitable options.

We will assume that the deliberator first selects the `drink-soda` option. As this option is to satisfy a new intrinsic goal, rather than a subgoal of a previous intention, a new intention stack is created. The end (goal) for the top intention frame of the stack is `quenched-thirst` and the means are given by the `drink-soda` plan.

As the first action in this plan is not primitive, no action is executed. Let us assume also that no external events occur on this cycle. Hence the only events in the event queue are the internal events corresponding to the creation of the new intention for the chosen option.

As the system has not succeeded in any of its goals nor discovered that any intentions are impossible, it posts the current intention status. This will result in the event `goal-add(have-soda)` being added to the event queue.[3]

The system then continues with the next cycle. Initially, the option generator will select the plan for getting soda from the refrigerator and this will, in turn, be adopted as an intention. The intention stack will

[3]Strictly, the system distinguishes between subgoals on the intention stack, or so-called *operand* goals, and the intrinsic goals contained in the goal data structure.

BEL	GOAL	INTEND	*done*	*succeeded*
have-glass	–	–	–	–
unchanged	quenched-thirst	–	–	goal-add(quenched-thirst)
unchanged	unchanged	{ have-soda; drink}	–	goal-add(have-soda)
¬ remove-soda	unchanged	–	open-fridge	open-fridge, goal-add(quenched-thirst)
unchanged	unchanged	{ drink}	open-tap	open-tap
quenched-thirst	–	–	drink	drink

Figure 3: Trace of Practical BDI Interpreter

thus be extended with a new intention frame corresponding to this plan. The interpreter will then execute the action of opening the refrigerator door, but at the next moment will discover that the soda is not present. It will thus be forced to drop its current intention. Finally, the initial intrinsic goal is reposted by post-intention-status.

On the next cycle, the option to drink water is selected, and the plan is finally completed successfully over the remaining cycles. The trace of the execution, including the goal-add events, is given in Figure 3.

5 CONCLUSIONS

The philosophical and logical foundations of rational agents have been studied by a number of authors [1, 3, 16, 18]. While most philosophical theories treat intentions as being reducible to beliefs and desires, Bratman argues that intentions play a significant and distinct role in practical reasoning and hence lays a philosophical foundation for BDI architectures. Cohen and Levesque [3] provided one of the first logical formalizations of intentions and the notion of commitment. We provided an alternative formalization by shifting the emphasis of future commitment from the definition of intentions to the process of intention maintenance. This allowed us to capture a number of different rational behaviors [16].

In parallel with the formalizations of BDI architectures, a number of system implementations that give primary importance to the mental attitudes of beliefs, goals, and intentions have emerged [5, 7].

However, none of the papers on formalizations of BDI architectures address the issue of how such systems could be implemented. Similarly, none of the system implementations try to relate their work to the formalizations. The only paper, to the knowledge of the authors, that attempts to bridge the gap between the two is by Pollack *et. al.* [14]. They describe the different modules of a BDI architecture and discuss the philosophical foundations for each of these modules. However, compared to the abstract interpreter presented here, their model is at a higher level of abstraction and is not useful as a practical system.

This paper has attempted to bridge the gap between the theoretical formalizations of BDI architectures, in particular our earlier formalization, and an implemented system based on the BDI architecture, the Procedural Reasoning System (PRS). The practical system architecture discussed in this paper is a simplified version of PRS, which has been used in a number of complex real-world applications.

Clearly, much of the interesting detail remains to be filled in. However, we hope that this paper has at least indicated the way in which suitable formalizations of rational behavior can be usefully employed as abstract specifications for building practical reasoning systems.

References

[1] M. E. Bratman. *Intentions, Plans, and Practical Reason.* Harvard University Press, Massachusetts, 1987.

[2] R. A. Brooks. A robust layered control system for a mobile robot. Technical Report 864, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1985.

[3] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3), 1990.

[4] E. A. Emerson and J. Srinivasan. Branching time temporal logic. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, pages 123–172. Springer-Verlag, Berlin, 1989.

[5] R. J. Firby. An investigation into reactive planning in complex domains. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 202–206, Seattle, Washington, 1987.

[6] M. P. Georgeff and F. F. Ingrand. Research on procedural reasoning systems. Final Report, Phase 2, for NASA Ames Research Center, Moffet Field, California, Artificial Intelligence Center, SRI International, Menlo Park, California, 1990.

[7] M. P. Georgeff and A. L. Lansky. Procedural knowledge. In *Proceedings of the IEEE Special*

Issue on Knowledge Representation, volume 74, pages 1383–1398, 1986.

[8] M. P. Georgeff and A. S. Rao. An Abstract Architecture for Rational Agents. Technical Report forthcoming, Australian Artificial Intelligence Institute, Carlton, Australia, 1992.

[9] M.P. Georgeff and F.F. Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, 1989.

[10] F. F. Ingrand, M. P. Georgeff, and A. S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert*, forthcoming, 1992.

[11] N. Karppinen, A. Lucas, M. Ljungberg, and P. Repusseau. Artificial Intelligence in Air Traffic Flow Management. Technical Report 16, Australian Artificial Intelligence Institute, Carlton, Australia, 1991.

[12] D. Kinny and M. P. Georgeff. Commitment and effectiveness of situated agents. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991.

[13] T. J. Laffey, P. A. Cox, J. L. Schmidt, S. M. Kao, and J. Y. Read. Real-time knowledge-based systems. *AI Magazine*, 9(1):27–45, 1988.

[14] M. E. Pollack, D. J. Israel, and M. E. Bratman. Towards an architecture for resource-bounded agents. Technical Report Technical Note 425, SRI International, Menlo Park, 1987.

[15] A. S. Rao and M. P. Georgeff. Intelligent real-time network management. In *Specialized Conference on Artificial Intelligence, Telecommunications, and Computer Systems (AVIGNON-90)*, 1990.

[16] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers, San Mateo, 1991.

[17] S. J. Rosenschein and L. P. Kaelbling. The synthesis of digital machines with provable epistemic properties. In J. Y. Halpern, editor, *Proceedings of the First Conference on Theoretical Aspects of Reasoning about Knowledge*, San Mateo, California, 1986. Morgan Kaufmann Publishers, Inc.

[18] E. Werner. Cooperating agents: A unified theory of communication and social structure. In L. Gasser and M. N. Huhns, editors, *Distributed Artificial Intelligence: Volume II*. Morgan Kaufmann Publishers, 1990.

Accessibility in Logics of Explicit Belief

James P. Delgrande
School of Computing Science,
Simon Fraser University,
Burnaby, B.C.,
Canada V5A 1S6

(604) 291-4335
jim@cs.sfu.ca

Abstract

In Kripke models of logics of implicit belief, different logics are obtained by altering the properties of the accessibility relation. In this paper I argue that no adequate account of (situation-based) explicit belief is possible based on an accessibility relation, or on a generalisation of an accessibility relation. If $B\alpha$ is read as "α is explicitly believed", then the problem is that the modality $\neg B$ conflates two distinct senses, that of not-believing and that of lack of belief. A specific explicit belief logic is proposed, and a new modal operator C is added to the logic to separate these senses, where $C\alpha$ is read "α is conceivable". Properties of the resultant logic are investigated and it is shown that the aforementioned difficulties do not arise.

1 INTRODUCTION

The epistemic notions of knowledge and belief have most commonly been modelled by means of possible worlds semantics, employing *Kripke Structures*. In a Kripke structure, the fundamental notions are those of a *possible world* and of an *accessibility relation* between worlds. A possible world is a complete and consistent state of affairs. The accessibility relation specifies those worlds that are accessible, or possible, according to what the agent whose beliefs are being modelled believes. In such approaches to *implicit belief* an agent is *logically omniscient*, in that it believes all logical consequences of its beliefs. Thus, if $L\alpha$ is used to assert that α is implicitly believed, then the sentence $L\alpha \wedge L(\alpha \supset \beta) \supset L\beta$ is valid.

Consequently, several approaches have been proposed to model systems of *explicit belief*. Such systems are intended to model finite agents or computers, where logical omniscience does not necessarily hold. In one of these approaches, a generalisation of possible worlds, called *situations*, is commonly adopted. The general methodology underlying this approach is to generalise Kripke structures to capture notions of explicit belief. In this paper I argue that *no* satisfactory account of situation-based explicit belief can be based directly on Kripke structures. The difficulty is that the accessibility relation leads to problems, as do generalisations of an accessibility relation. This is unfortunate, since in possible worlds approaches different logics of implicit belief are obtained by varying the properties of the accessibility relation. I suggest that the problem arises because in systems of explicit belief a negated belief is effectively used in two distinct senses.

In logics of implicit belief, beliefs must be consistent. This is a direct consequence of the fact that worlds, which are used to model beliefs, are complete and consistent state of affairs. Thus if the sentence $\neg L\alpha$ is true at a world then $L\alpha$ fails to be true at that world. However, in a logic of explicit belief this is not the case, and if $\neg B\alpha$ has true support at a situation, it may also be the case that $B\alpha$ has true support.[1] This leads to implausible properties, regardless of how an accessibility relation is defined (or implicitly adopted) in these logics. I argue that an additional modality is required in order that these senses be distinguished. This second modal operator is intended to capture the notion that something is inconceivable, written $\neg C\alpha$. This is a somewhat stronger notion that simple non-belief, represented by $\neg B\alpha$. A specific logic is developed, and the resultant framework is argued to be adequate for the full representation of explicit belief.

The next section provides a brief introduction to logics of implicit and explicit belief. Section 3 lists adequacy criteria for explicit belief logics, and shows that no extant system fulfills these requirements. Also I argue in Section 4 that no generalisation of these systems, *can* fulfill these requirements. Section 5 describes the proposed solution, while Section 6 provides a brief discussion.

[1] These terms are gone into in more detail in the following sections.

2 OVERVIEW OF IMPLICIT AND EXPLICIT BELIEF

The following subsections summarise recent work in implicit and explicit belief. For more thorough introductions, [McA88] surveys systems of implicit and explicit belief, while [HM92] provides a comprehensive introduction to implicit belief, with emphasis on decision problems.

2.1 IMPLICIT BELIEF

A central problem in the modeling of knowledge and belief is that of determining the full set of beliefs intrinsic in a base set of beliefs of some agent. This question has been addressed most frequently in formal models of belief by means of a *possible worlds* semantics. In such approaches an agent's beliefs are modelled by a set of possible worlds that specify, roughly, how the world could be, assuming that the agent's base set of beliefs was in fact true. In these systems an agent is logically omniscient in that it believes all logical consequences of its beliefs. Omniscience is easily shown by an informal argument: if an agent believes some set of sentences Γ, then every sentence in Γ must be true at every world modelling the agent's beliefs. But since worlds are complete and consistent, every logical consequence of Γ must also be true at every world. Hence, since a logical consequence is true at every world, by definition it is implicitly believed. Similar arguments show that an agent also believes all logically valid sentences, and if an agent believes a contradiction, then it believes every sentence. In order not to seem wholly unrealistic, implicit belief is usually interpreted as either modelling an idealisation of an agent, or else as specifying what is implicit in an agent's beliefs. The initial work in this area is [Hin62]; representative work in Artificial Intelligence includes [Kon84, Lev84a, Moo80].

Formally such belief is modelled by a *Kripke structure* $\mathcal{M} =< W, R, P >$ where W is a set of possible worlds, R is a binary relation on W giving, for each world, the set of accessible worlds, and P is a mapping from the set of atomic sentences $\mathbf{P}$ onto subsets of W, specifying the worlds at which each atomic sentence is true. The truth of sentences is given by a straightforward recursive definition. The interesting case is for an *implicit belief* $L\alpha$, which is true at w iff α is true at all accessible worlds.

More formally, I will assume throughout that we have a propositional language, $\mathcal{L}$, for expressing explicit and implicit belief. This language is formed from a set of atomic sentences $\mathbf{P} = \{p_0, p_1, ...\}$, together with the standard connectives $\neg$ and $\wedge$ for negation and conjunction, and operators B and L for explicit and implicit belief respectively. Disjunction and material implication are introduced by the usual definitions. Sentences of $\mathcal{L}$ are specified by the expected recursive definition. Lower case Greek letters α, β, ... stand for arbitrary sentences of $\mathcal{L}$. The symbolism $\mathcal{M}, w \models \alpha$ asserts that α is *true* at world w in model $\mathcal{M}$. If α is true at every world and in every model, α is valid, written $\models \alpha$. We obtain:

Definition 2.1

1. $\mathcal{M}, w \models p_i$ iff $w \in P(p_i)$.
2. $\mathcal{M}, w \models \neg\alpha$ iff $\mathcal{M}, w \not\models \alpha$.
3. $\mathcal{M}, w \models \alpha \wedge \beta$ iff $\mathcal{M}, w \models \alpha$ and $\mathcal{M}, w \models \beta$.
4. $\mathcal{M}, w \models L\alpha$ iff for every w' such that $R(w, w')$, we have $\mathcal{M}, w' \models \alpha$.

There is no universally-accepted logic of implicit belief; rather, depending on one's requirements, there are more (or less) suitable logics. If we begin with a completely unconstrained accessibility relation, we obtain the base logic of implicit belief, K [Che80]. A proof-theoretic characterisation of K is given by adding to classical propositional logic the axiom

$$L(\alpha \supset \beta) \supset L\alpha \supset L\beta$$

and rule of inference

$$\text{From } \vdash \alpha \text{ infer } \vdash L\alpha.$$

By imposing various conditions on the accessibility relation, other logics are obtained. Thus for example, if the accessibility relation is reflexive, then $L\alpha \supset \alpha$ is valid. This formula, which states that a believed sentence is in fact true, is usually taken as distinguishing knowledge from simple belief, which may not be true.[2] If the accessibility relation is transitive, then $L\alpha \supset LL\alpha$ is valid; this relation of positive introspection states that if an agent believes something then it believes that it believes it. If the accessibility relation is euclidean [3], then the relation of negative introspection, $\neg L\alpha \supset L\neg L\alpha$, is valid. Note that if an accessibility relation is reflexive, transitive, and euclidean, then it in fact constitutes an equivalence relation.

2.2 EXPLICIT BELIEF

The notion of implicit belief leaves open the issue of what a resource-bounded, finite agent might believe or actively hold as true. A variety of systems of *explicit belief* have subsequently been developed to address this issue. In these systems an agent is not necessarily logically omniscient, and so if it believes α and $\alpha \supset \beta$ then it may not necessarily believe β. However, given that an agent's explicit beliefs may not be closed under

[2] The distinction between knowledge and belief will not concern us here, and I will use the term "belief" throughout.

[3] R is euclidean if $R(w_1, w_2)$ and $R(w_1, w_3)$ implies that $R(w_2, w_3)$

(classical) logical consequence, it is unclear just what explicit beliefs (if any) may follow from a base set.

Systems of explicit belief generally belong to one of two broad approaches, the *syntactic* and *situational* approaches. In the syntactic approach, an agent's explicit beliefs are modelled by a set of sentences. Approaches based on these intuitions include [Ebe74, FH85, Haa86, Kon84, MH79]. In the most basic of these approaches, an agent explicitly believes a sentence just when it appears in the set. Logical omniscience is clearly blocked in these systems, since a logical consequence of a set of sentences may not appear in that set. Also, this approach represents the least constrained conception of explicit belief. However this approach is perhaps too fine-grained, in that the set of sentences which are or are not believed may be quite arbitrary. Thus, believing that "Mary is a scholarship recipient and John is a teacher" may not entail that "John is a teacher and Mary is a scholarship recipient" is believed. However, arguably, these sentences can been seen to represent the same belief, and the ordering of the conjuncts is an irrelevant syntactic commitment. While rules could be introduced to ensure that, for example, individual conjuncts of a conjunction are believed, such rules are difficult to justify without an underlying semantic theory.

The situational approach extends possible worlds semantics, and attempts to provide a semantically well-motivated account of explicit belief. Here an agent's explicit beliefs are modelled by a set of partial possible worlds or *situations*. A situation may *support* either the truth or the falsity of a sentence, or both the truth and falsity, or neither.[4] Again an agent may not be logically omniscient in these systems; for example it may not believe all the (classical) consequences of its beliefs.

The original work in Artificial Intelligence in this area is [Lev84b]. In this approach a model is given by $\mathcal{M} =< \mathcal{S}, \mathcal{B}, \mathcal{T}, \mathcal{F} >$ where $\mathcal{S}$ is the set of all situations, $\mathcal{B}$ is the set of situations which could be the case given what is believed, and $\mathcal{T}$ and $\mathcal{F}$ are functions from the set of atomic sentences P onto subsets of $\mathcal{S}$. $\mathcal{T}$ specifies those atomic sentences that have their truth supported at each situation, while $\mathcal{F}$ specifies those atomic sentences that have their falsity supported. Atomic sentence p_i has its truth supported at situation s in model $\mathcal{M}$ iff $s \in \mathcal{T}(p_i)$. Similarly p_i has its falsity supported at s iff $s \in \mathcal{F}(p_i)$. The fact that p_i could have neither its truth nor falsity supported at s reflects the intuition that p_i may have nothing to so with that particular situation. The fact that p_i could have both its truth and falsity supported at s, in an *incoherent* situation, reflects the fact that unfortunate as the fact is, contradictions must be expected and lived with in a large knowledge base. Support of truth and of falsity of compound sentences is given by a straightforward recursive definition. The set W of worlds is defined to be the set of complete and consistent situations. We obtain:

Definition 2.2

1. $\mathcal{M}, s \models_T p_i$ iff $s \in \mathcal{T}(p_i)$.
   $\mathcal{M}, s \models_F p_i$ iff $s \in \mathcal{F}(p_i)$.
2. $\mathcal{M}, s \models_T \neg\alpha$ iff $\mathcal{M}, s \models_F \alpha$.
   $\mathcal{M}, s \models_F \neg\alpha$ iff $\mathcal{M}, s \models_T \alpha$.
3. $\mathcal{M}, s \models_T \alpha \wedge \beta$ iff $\mathcal{M}, s \models_T \alpha$ and $\mathcal{M}, s \models_T \beta$.
   $\mathcal{M}, s \models_F \alpha \wedge \beta$ iff $\mathcal{M}, s \models_F \alpha$ or $\mathcal{M}, s \models_F \beta$.
4. $\mathcal{M}, s \models_T \alpha \vee \beta$ iff $\mathcal{M}, s \models_T \alpha$ or $\mathcal{M}, s \models_T \beta$.
   $\mathcal{M}, s \models_F \alpha \vee \beta$ iff $\mathcal{M}, s \models_F \alpha$ and $\mathcal{M}, s \models_F \beta$.
5. $M, s \models_T B\alpha$ iff for every $t \in \mathcal{B}$, $M, t \models_T \alpha$.
   $M, s \models_F B\alpha$ iff $M, s \not\models_T B\alpha$.

In this approach, both *support of truth* ($\models_T$) and *truth* ($\models$) are defined. The former is defined at all situations, as given above; the latter is defined only at worlds, as with implicit belief. An agent may not be logically omniscient in this approach. For example, the sentences $B\alpha$, $B(\alpha \supset \beta)$, and $\neg B\beta$ are simultaneously satisfiable and so an agent's beliefs are not closed under implication. In addition the sentences $B(\alpha \wedge \neg\alpha)$ and $\neg B\beta$ are simultaneously satisfiable, and so inconsistent beliefs can be held without everything being believed. Also $\neg B(\alpha \vee \neg\alpha)$ is satisfiable; hence a valid sentence need not be explicitly believed.

If the semantics were specified in terms of an accessibility relation, this relation would be transitive and euclidean, but not necessarily reflexive: for situation s, there is a set of mutually accessible situations $\mathcal{B}$, but it may be that $s \notin \mathcal{B}$. The approach also defines an implicit belief operator. Since it is not germane to the point at hand (viz. accessibility with respect to explicit belief), I do not include it in the above definition, nor is it dealt with in the following subsections. Adding an implicit belief operator to the subsequent development however is straightforward.

As [FH85] points out, there are difficulties with this approach. Most importantly, the sentence

$$B\alpha \wedge B(\alpha \supset \beta) \supset B(\beta \vee (\alpha \wedge \neg\alpha))$$

is valid. Thus if $B\alpha$ and $B(\alpha \supset \beta)$ are true then so is $B(\beta \vee (\alpha \wedge \neg\alpha))$. If $\neg B\beta$ is also true then semantically we are committed to the existence of an incoherent situation wherein both α and $\neg\alpha$ are supported. But this seems unreasonable: in general, lack of logical omniscience seems to have little to do with incoherent situations.

[Lev85] addresses this problem by extending the approach so that an agent's explicit beliefs are modelled

[4] A situation here then may be seen as a restricted version of a situation in [BP83].

by a set of sets of situations $\{\mathcal{C}_1, \mathcal{C}_2, \ldots\}$. Support for explicit belief is now given by:

$\mathcal{M}, s \models_T B\alpha$ iff for some $\mathcal{C}_i$ and every $t \in \mathcal{C}_i$,

we have $\mathcal{M}, t \models_T \alpha$.

Informally each $\mathcal{C}_i$ models a particular "state of mind" or "belief context".[5] While this approach provides an alternative to incoherent situations as a means of blocking omniscience, it leaves the notion of belief context largely unanalysed; moreover it does not allow iterated belief modalities. These issues are addressed in [Del92], wherein $B\alpha$ has true support at situation s just when the *explicit proposition* represented by α appears among a set, $N_T(s)$, of explicit propositions associated with s. This set $N_T(s)$ then consists of the set of propositions explicitly believed at situation s. A second set $N_{\mathcal{F}}(s)$ specifies those propositions which are not believed at s. This approach does not address the concerns of the present paper however, in that the notion of an accessibility relation is not employed.

Gerhard Lakemeyer has addressed numerous issues in explicit belief by considering various extensions to [Lev84b]. [Lak87] allows reasoning about explicit beliefs at any level (that is, nesting of belief operators) provided that all beliefs are at the same level. Thus for example, not only is $B(\alpha \wedge \beta) \supset B\beta$ valid, but so is $BB(\alpha \wedge \beta) \supset BB\beta$. This is accomplished by introducing two accessibility relations that hold between situations: one relation is used to determine the truth of unnegated beliefs, while the other is used for negated beliefs. The rationale for these two relations is that an agent may use different sets of situations to confirm or disconfirm a belief. [Lak86] addresses issues in a first-order logic of explicit belief, while [LL88] considers embedding additional modal operators in the approach of [Lev84b]. However the logics of these papers do not address the aforementioned problems and so block implication in explicit beliefs via incoherent situations.

3 ADEQUACY CRITERIA FOR EXPLICIT BELIEF

In possible worlds semantics the base logic K is modelled by the full set of Kripke structures; further logics are obtained by constraining the accessibility relation. It might be thought that we could do the same thing with explicit belief, and so provide a base logic (conforming to some base semantics), and from this specify further logics by restricting an accessibility relation. However this appears to not be the case, and I argue that no adequate basis for explicit belief is possible based directly on the notions of situation and accessibility relation. In order to demonstrate this, we need to first specify what would constitute an "adequate basis" for explicit belief. Arguably, any semantic basis for explicit belief should satisfy at least the following criteria. [6]

1. Relations involving iterated belief modalities are specifiable.
2. $\{B\alpha, B(\alpha \supset \beta), \neg B\beta\}$ is satisfiable.
3. $\{B\alpha, B(\alpha \supset \beta), \neg B(\beta \vee (\alpha \wedge \neg\alpha))\}$ is satisfiable.
4. $\{B\alpha, B\beta, \neg B(\alpha \wedge \beta)\}$ is satisfiable.
5. Three degrees of coherence in explicit beliefs are specifiable:
   (a) $B(\alpha \wedge \neg\alpha)$ is satisfiable.
   (b) $\neg B(\alpha \wedge \neg\alpha)$ is valid; $\{B\alpha, B\neg\alpha\}$ is satisfiable.
   (c) $B\alpha \supset \neg B\neg\alpha$ is valid.

The first criterion states that one should be able to stipulate, for example, that beliefs are closed under positive or negative introspection, and so $B\alpha \supset BB\alpha$ and $\neg B\alpha \supset B\neg B\alpha$, respectively, are valid. [Lev84b] does not address issues of meta-beliefs; however, since in the semantics of [Lev84b] we have that either $\mathcal{M}, s \models_T B\alpha$ or $\mathcal{M}, s \models_T \neg B\alpha$, it follows that in any obvious extension to the system an agent will be omniscient with respect to its meta-beliefs. Similar remarks apply to [Lev85].

[Lak87] addresses this difficulty by developing an explicit belief logic incorporating two accessibility relations, one for positive explicit beliefs, and another for negated explicit beliefs. This permits a flexible specification of iterated modalities. However the accessibility relations are constrained to coincide at worlds, and so, as is discussed below, the other adequacy conditions are not met. Note that the two accessibility relations *must* coincide at worlds; otherwise the resultant system is inconsistent.

The second criterion states that an agent's beliefs are not closed under logical consequence; this indeed is the case in all cited works. However in the above cited works, with the exception of [Lev85], $B\alpha \wedge B(\alpha \supset \beta) \supset B(\beta \vee (\alpha \wedge \neg\alpha))$ is valid, violating the third criterion. This last formula effectively requires that, if an agent's beliefs are not closed under logical consequence, then one of the situations the agent believes possible is incoherent, in which $\alpha \wedge \neg\alpha$ has true support. But this seems implausible: if an agent holds $B\alpha$ and separately holds $B(\alpha \supset \beta)$ then it is unreasonable to require that the only way the inference $B\beta$ is blocked is via an incoherent situation. Arguably the agent may not have "connected" these two beliefs and so may not hold $B\beta$. The fact that β is not believed then would, via this line of reasoning, have nothing to do with incoherent situations.

[5] [FH85] presents a similar approach founded on intuitions for "weak" and "strong" implicit belief.

[6] While there may be other criteria, these are pertinent to the discussion at hand.

This point is expressed in the fourth criterion, where we require that an agent be able to hold two separate beliefs, without necessarily holding their conjunction. It is straightforward to show that the third condition effectively amounts to the fourth. The fourth condition states that if α and β are explicitly believed then their conjunction may not be believed. Arguably this is how things should be: if an agent has two beliefs, again, it may not have explicitly "connected" them, and so may not explicitly hold their conjunction. However, in the previously-cited approaches (with the exception of [Lev85]) $B\alpha \wedge B\beta \equiv B(\alpha \wedge \beta)$ is valid.

The last criterion says that there are three "degrees of coherence" of beliefs, and that, by suitably constraining the semantics, an approach should allow one to distinguish these degrees. First, an agent may have incoherent beliefs (of the form $B(\alpha \wedge \neg\alpha)$); second, an agent may not have incoherent beliefs, but may have conflicting beliefs (of the form $B\alpha$ and $B\neg\alpha$); and third, an agent will not have conflicting beliefs, and so if it believes α then it will not believe $\neg\alpha$.[7]

The preceding has argued that no extant system of explicit belief has satisfied the adequacy criteria set out at the beginning of this section. It is straightforward to show that no situation-based semantics for explicit belief incorporating an accessibility relation *can* satisfy the adequacy criteria. As mentioned, the standard notion of an accessibility relation cannot be imported wholesale into situation-based semantics for explicit belief without reintroducing omniscience via meta-beliefs. In addition the obvious extension to situation-based semantics also violates the fourth criterion. To see this, assume to the contrary that we have some situation-based semantics based on an accessibility relation. If Bp_1 and Bp_2 are both true at world w then, by the definition of support of truth, p_1 and p_2 have true support at all accessible situations. But then $p_1 \wedge p_2$ has true support at all accessible situations; hence $B(p_1 \wedge p_2)$ is true at the original world. But this violates the fourth of the adequacy criteria. A similar argument shows the third criterion is also violated.

The preceding example however points the way to a possible solution. Arguably Bp_1 and Bp_2 should not lead to $B(p_1 \wedge p_2)$ because, informally, p_1 and p_2 may have nothing to do with each other. For example, I may explicitly believe that "I am presently working at my desk" and "the muffler on my car needs replacing", but I may not believe that "I am presently working at my desk and the muffler on my car needs replacing". In the phrasing of [Lev85] an agent may believe one thing in one "belief context" or "frame of mind", and another thing in another; there is no a priori reason then that the agent has "connected" these belief contexts. Semantically we would want to allow the truth of Bp_1 to depend on one set of situations while the truth of Bp_2 could depend on another. Thus $B(p_1 \wedge p_2)$ need not be held, since $p_1 \wedge p_2$ may not have true support at whatever the appropriate set of situations is for that belief. Thus we can extend the notion of accessibility so that, for belief $B\alpha$, the set of accessible situations depends not just on a situation (as before), but also in some fashion on α.

[7] Note however that while previous situation-based approaches rely on incoherent situations to block logical omniscience, the discussion in Sections 4 and 5 does not depend on the presence of incoherent situations; the same points apply to situation-based approachs in which there are *no* incoherent situations.

4 PROPOSITION-BASED ACCESSIBILITY RELATIONS

4.1 ALTERNATIVE APPROACHES

The natural generalisation of an accessibility relation is to have the set of accessible situations depend in part on the proposition expressed by a sentence α. That is, in possible worlds approaches, it is fairly common practice [Che80, Sta87] to identify the proposition (or intension) expressed by a sentence α with the set of possible worlds in which α is true. If α and β are true at precisely the same worlds in a model then α and β represent the same proposition. In such approaches then a sentence α is taken as denoting some proposition, and it is this proposition that is the object of belief. This idea is extended here to that of *explicit propositions*, where the explicit proposition expressed by α is identified with the set of situations in which α has true support; this is written as $\|\alpha\|_S^{\mathcal{M}}$. Thus we have:

Definition 4.1

$$\|\alpha\|_S^{\mathcal{M}} = \{s \in \mathcal{S} \mid \mathcal{M}, s \models_T \alpha\}.$$

For belief $B\alpha$, the set of accessible situations depends on the situation s and on the explicit proposition (henceforth, proposition) expressed by α; this will be given by a function $f(s, \|\alpha\|_S^{\mathcal{M}})$. A model is now a tuple $\mathcal{M} = < \mathcal{S}, f, \mathcal{T}, \mathcal{F} >$ where $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{F}$ are as before, and f is a function from situations and sets of situations (representing explicit propositions) onto sets of situations. Again, the function f is a basic entity in the semantic theory, generalising the accessibility relation of Kripke structures.

True support of an explicit belief can be defined as:

$$M, s \models_T B\alpha \text{ iff for every } t \in f(s, \|\alpha\|_S^{\mathcal{M}}), \text{ we have } M, t \models_T \alpha,$$

or equivalently,

$$M, s \models_T B\alpha \text{ iff } f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}.$$

However this semantics is overly weak, in that the arguably-desirable relation $B(\alpha \wedge \beta) \supset B\alpha$ is not valid. To obtain this notion of specificity between propositions we also require that:

If $\|\alpha\|_S^{\mathcal{M}} \subseteq \|\beta\|_S^{\mathcal{M}}$ then $f(s, \|\beta\|_S^{\mathcal{M}}) \subseteq f(s, \|\alpha\|_S^{\mathcal{M}})$.

A rationale for this relation is as follows. Consider again the sentences "I am presently working at my desk" and "I am presently working at my desk and the muffler on my car needs replacing". The events described in the second sentence portray a broader or more general situation than does the first. That is, the first sentence deals only with my office, while the second includes my office and car. Thus if true support of α at a situation requires true support of β at that situation, then α, being more specific than β, describes a narrower state of affairs. Based on this observation, we may require that if an proposition is believed then so is any subsuming, or more general, proposition.

In the resultant semantics $B(\alpha \wedge \beta) \supset B\alpha$ is valid, since $\|\alpha \wedge \beta\|_S^{\mathcal{M}} \subseteq \|\alpha\|_S^{\mathcal{M}}$, and so if $f(s, \|\alpha \wedge \beta\|_S^{\mathcal{M}}) \subseteq \|\alpha \wedge \beta\|_S^{\mathcal{M}}$ then $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$. Notably $B\alpha \wedge B\beta \supset B(\alpha \wedge \beta)$ is not valid. Adequacy criteria 1.–4. are met; this is discussed once the full system has been presented.

However at this point we have not yet defined support of the falsity of an explicit belief (i.e. $M, s \models_F B\alpha$). The difficulty is that, while it is clear how positive explicit belief should be defined, it is unclear how negated explicit belief should be defined. At worlds there is no problem, since we must have $M, w \models \neg B\alpha$ iff $M, w \not\models B\alpha$. At situations we have $M, s \models_T \neg B\alpha$ iff $M, s \models_F B\alpha$, and so we need to define this last relation.

There seems to be four possibilities for $M, s \models_F B\alpha$. First, the set of accessible situations may depend on the explicit proposition expressed by α, or it may depend on that expressed by $\neg\alpha$.[8] Second, given a set of accessible situations, we may require either that a situation support the falsity of α or else that a situation simply fail to support the truth of α.

We have the possibilities:

1. $M, s \models_F B\alpha$ iff $M, s \not\models_T B\alpha$
   (and so $f(s, \|\alpha\|_S^{\mathcal{M}}) \not\subseteq \|\alpha\|_S^{\mathcal{M}}$).
2. $M, s \models_F B\alpha$ iff $f(s, \|\neg\alpha\|_S^{\mathcal{M}}) \cap \|\neg\alpha\|_S^{\mathcal{M}} \neq \emptyset$.
3. $M, s \models_F B\alpha$ iff $f(s, \|\neg\alpha\|_S^{\mathcal{M}}) \not\subseteq \|\alpha\|_S^{\mathcal{M}}$.
4. $M, s \models_F B\alpha$ iff $f(s, \|\alpha\|_S^{\mathcal{M}}) \cap \|\neg\alpha\|_S^{\mathcal{M}} \neq \emptyset$.

Some of these possibilities are *intuitively* implausible. The first three possibilities can also be shown to have undesirable *formal* properties. The fourth, while reasonable, requires augmenting the approach with another modal operator to be fully useful or expressive.

The intuition underlying the first possibility is that of classical modal logics: that falsity of an explicit belief corresponds to non-(support of) truth. We have that either $\mathcal{M}, s \models_T B\alpha$ or $\mathcal{M}, s \models_T \neg B\alpha$, and so, as with [Lev84b], an agent is omniscient with respect to its meta-beliefs.

For the second possibility, the intuition is that the set of situations relevant to falsity of $B\alpha$ is determined by the negation of α, and a belief has false support just when there is a situation that supports the falsity of the sentence. Since $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$ implies that $f(s, \|\alpha\|_S^{\mathcal{M}}) \cap \|\alpha\|_S^{\mathcal{M}} \neq \emptyset$ (assuming that $f(s, \|\alpha\|_S^{\mathcal{M}}) \neq \emptyset$), we obtain that $B\alpha \supset \neg B\neg\alpha$ is valid. Hence, in non-trivial cases this possibility violates the last of the adequacy criteria.

The intuition behind the third possibility is that the set of situations relevant to falsity of $B\alpha$ is again determined by the negation of α; however, the belief has false support just when there is an accessible situation that fails to support the truth of the sentence. This approach then extends [Lak87]. However it violates the fifth criterion. Consider the situation where we wish to disallow incoherent beliefs. For $\neg B(\alpha \wedge \neg\alpha)$ to be valid we require that $f(s, \|\alpha \vee \neg\alpha\|_S^{\mathcal{M}}) \cap \|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}} = \emptyset$. Thus $\neg B(\alpha \wedge \neg\alpha)$ is valid in the class of models in which $\|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}} = \emptyset$.[9] Hence we exclude incoherent situations in the semantics. $\neg B(\alpha \wedge \neg\alpha)$ is now valid; but since $\|\alpha\|_S^{\mathcal{M}} \cap \|\neg\alpha\|_S^{\mathcal{M}} = \emptyset$, we have that:

If $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$ then $f(s, \|\alpha\|_S^{\mathcal{M}}) \not\subseteq \|\neg\alpha\|_S^{\mathcal{M}}$.

This means that $B\alpha \supset \neg B\neg\alpha$ is valid also, violating the fifth criterion. Hence this possibility is too strong.

The last alternative appears to be the most intuitively and formally appealing of the four. Essentially it states that $\neg B\alpha$ has true support just when there is a situation, among those relevant to belief in α, that supports the falsity of α. Note that, in particular, we use the same set of situations to determine support of truth or falsity in a sentence α. This system is explored in the next subsection.

4.2 A PROPOSITION-BASED LOGIC OF EXPLICIT BELIEF

This subsection discusses the system where support of truth is given by:

$M, s \models_T B\alpha$ iff $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$.

$M, s \models_F B\alpha$ iff $f(s, \|\alpha\|_S^{\mathcal{M}}) \cap \|\neg\alpha\|_S^{\mathcal{M}} \neq \emptyset$.

[8] The other plausible possibility, of having the set of accessible situations depend on $\|\alpha \vee \neg\alpha\|_S^{\mathcal{M}}$, violates criterion 5b in that $B\alpha \wedge B\neg\alpha \equiv B(\alpha \wedge \neg\alpha)$ is valid.

[9] Note having $f(s, \|\alpha \vee \neg\alpha\|_S^{\mathcal{M}}) = \emptyset$ is not a satisfactory option, since it would mean that if inconsistencies are not explicitly believed then $\alpha \vee \neg\alpha$ is only satisfied in the trivial case.

It is worth emphasising that *support of truth* is distinct from *truth*, which is defined only at worlds and in the expected manner:

$M, w \models B\alpha$ iff $f(w, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$.

$M, w \models B\alpha$ iff $M, w \not\models B\alpha$.

While truth (and so validity) is defined only at worlds, the definition of truth for explicit belief is given in terms of support of truth.

This approach has the rather nice properties that:

1. If $\|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}} = \emptyset$ then: if $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$ then $f(s, \|\alpha\|_S^{\mathcal{M}}) \cap \|\neg\alpha\|_S^{\mathcal{M}} = \emptyset$.
2. If $\|\alpha \vee \neg\alpha\|_S^{\mathcal{M}} = \mathcal{S}$ then: if $f(s, \|\alpha\|_S^{\mathcal{M}}) \cap \|\neg\alpha\|_S^{\mathcal{M}} = \emptyset$ then $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$.

The first item means that, if $\|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}} = \emptyset$, then if $\mathcal{M}, s \models_T B\alpha$ then $\mathcal{M}, s \not\models_F B\alpha$. The second item gives the converse. Consequently if $\mathcal{S} = W$ (and so $\|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}} = \emptyset$ and $\|\alpha \vee \neg\alpha\|_S^{\mathcal{M}} = \mathcal{S}$ for every α) we obtain that $\mathcal{M}, s \models_T B\alpha$ iff $\mathcal{M}, s \not\models_F B\alpha$. Thus support of truth and of falsity in this case are equivalent to classical notions of truth and falsity (and we in fact obtain the modal logic EM [Che80, p. 236]).

For the first adequacy criterion, relations among explicit belief modalities can be specified in a manner analogous to possible worlds semantics. Thus, for example, the statement $B\alpha \supset \alpha$ is valid in the class of models in which $s \in f(s, \|\alpha\|_S^{\mathcal{M}})$ for every situation and explicit proposition. That is, $B\alpha \supset \alpha$ has true support at situation s iff $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$ implies that α has true support at s. This in turn means that $s \in f(s, \|\alpha\|_S^{\mathcal{M}})$, corresponding to reflexivity. Positive and negative introspection is handled in a straightforward, if slightly more complex, fashion.

Adequacy criteria 2.–4. clearly are met, in that support of the truth of $B\alpha$ and $B\beta$ can depend on different sets of situations whenever $\|\alpha\|_S^{\mathcal{M}} \neq \|\beta\|_S^{\mathcal{M}}$. Since $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq f(s, \|\alpha \wedge \beta\|_S^{\mathcal{M}})$, we have that $B(\alpha \wedge \beta) \supset B\alpha \wedge B\beta$, but not the converse.

The final adequacy criterion is also met, although not perhaps in an entirely satisfactory manner. Assume for the moment that $f(s, \|\alpha\|_S^{\mathcal{M}}) \neq \emptyset$ for every s and α. We obtain:

1. $B(\alpha \wedge \neg\alpha)$ is satisfiable.
2. $\neg B(\alpha \wedge \neg\alpha)$ is valid in the class of models in which $\|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}} = \emptyset$, while $\{B\alpha, B\neg\alpha\}$ remains satisfiable.
3. $B\alpha \supset \neg B\neg\alpha$ is valid in the class of models in which $\|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}} = \emptyset$ and $f(s, \|\alpha\|_S^{\mathcal{M}}) = f(s, \|\neg\alpha\|_S^{\mathcal{M}})$.

These semantic restrictions appear reasonable and intuitive; however the approach is not without difficulties. The problem is that, while we can state that a belief receives true support or false support, we cannot assert that a belief *fails* to receive true support. That is, at worlds, if $\mathcal{M}, w \models \neg B\alpha$ then $\mathcal{M}, w \not\models B\alpha$. However, at situations, support of truth and of falsity are independent[10] and so if $\mathcal{M}, s \models_T \neg B\alpha$ then we may or may not have $\mathcal{M}, s \not\models_T B\alpha$. This in turn, informally, means that we cannot "assert" that $f(w, \|\alpha\|_S^{\mathcal{M}}) \not\subseteq \|\alpha\|_S^{\mathcal{M}}$, or, as a special case, that $f(w, \|\alpha\|_S^{\mathcal{M}}) \neq \emptyset$. This has two consequences, first, that we cannot axiomatically provide a seriality condition for f, and, second, that we cannot assert that a belief fails to receive true support. These points are explored in the remainder of this section.

Consider first the case where we wish to exclude incoherent beliefs. If we want an agent's beliefs to be coherent, then first we would want $\neg B(\alpha \wedge \neg\alpha)$ to be valid, and so $f(w, \|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}}) \not\subseteq \|\alpha \vee \neg\alpha\|_S^{\mathcal{M}}$ for every $w \in W$. Consequently $f(w, \|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}}) \neq \emptyset$. This then amounts to a seriality condition for f at incoherent propositions and at worlds.

If we want additionally that an agent's meta-beliefs be coherent, and so $\neg BB(\alpha \wedge \neg\alpha)$ is valid, then first we would want $f(w, \|B(\alpha \wedge \neg\alpha)\|_S^{\mathcal{M}}) \not\subseteq \|B(\alpha \wedge \neg\alpha)\|_S^{\mathcal{M}}$ for every $w \in W$. Now, we don't necessarily have $\|B(\alpha \wedge \neg\alpha)\|_S^{\mathcal{M}} = \emptyset$, since the preceding discussion, wherein $\neg B(\alpha \wedge \neg\alpha)$ is valid, implies only that $\|B(\alpha \wedge \neg\alpha)\|_S^{\mathcal{M}} \cap W = \emptyset$.[11] So if $f(w, \|B(\alpha \wedge \neg\alpha)\|_S^{\mathcal{M}}) \not\subseteq \|B(\alpha \wedge \neg\alpha)\|_S^{\mathcal{M}}$ then for some $s \in f(w, \|B(\alpha \wedge \neg\alpha)\|_S^{\mathcal{M}})$ we have $\mathcal{M}, s \not\models_T B(\alpha \wedge \neg\alpha)$, and so $f(s, \|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}}) \not\subseteq \|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}}$. Hence from the above, $\neg BB(\alpha \wedge \neg\alpha)$ is valid in the class of models in which $\|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}} = \emptyset$ and $f(s, \|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}}) \neq \emptyset$. This then amounts to a seriality condition, but restricted to incoherent propositions.

Unfortunately, it appears to be the case that there is *no* axiomatic specification of seriality within the present framework. Consider first logics of implicit belief, where a formula of the form $L\alpha \supset \neg L\neg\alpha$ is valid in all serial models: if $L\alpha$ is true, then α is true at all accessible worlds; since $\neg L\neg\alpha$ is also true, then it is not the case that at all accessible worlds $\neg\alpha$ is true. Since worlds are consistent, this in turn implies that there *is* an accessible world. However we can't simply adopt $B\alpha \supset \neg B\neg\alpha$ as a relation here to obtain seriality because it is too strong, in that it violates the

[10] with the exception of incoherent beliefs, as discussed below.

[11] $\neg BB(\alpha \wedge \neg\alpha)$ is valid, clearly, in the set of models in which $\|B(\alpha \wedge \neg\alpha)\|_S^{\mathcal{M}} = \emptyset$ and $f(w, \|B(\alpha \wedge \neg\alpha)\|_S^{\mathcal{M}}) \neq \emptyset$. However, this does not give us a *tight* characterisation of the class of models in which incoherent meta-beliefs are not held. If we were interested in *any* class of models in which the preceding relation held, then we could simply point to the class of models in which $\|\alpha\|_S^{\mathcal{M}} = \emptyset$ for every α and be done with it. Here then we would like a precise characterisation.

fifth adequacy criterion.

What this all amounts to is that, with the exception of incoherent propositions, we cannot axiomatically "enforce" seriality. This is perhaps best seen by considering the introduction of a designated atomic sentence $\top$ that receives true support at every situation. Thus $\|\top\|^{\mathcal{M}}_S = \mathcal{S}$, and since $f(s, \|\top\|^{\mathcal{M}}_S) \subseteq \|\top\|^{\mathcal{M}}_S$, $B\top$ has true support at every situation. However there is nothing preventing $f(s, \|\top\|^{\mathcal{M}}_S) = \emptyset$. If we could axiomatically characterise the class of models in which $f(s, \|\top\|^{\mathcal{M}}_S) \neq \emptyset$ then we would have a full characterisation of seriality, since by specificity $f(s, \|\top\|^{\mathcal{M}}_S) \subseteq f(s, \|\alpha\|^{\mathcal{M}}_S)$ for every α. It might be thought that we would obtain seriality in the class of models in which $B\top \supset \neg B\neg\top$ is valid. However this is not so: semantically we obtain that if $f(s, \|\top\|^{\mathcal{M}}_S) \subseteq \top$ then $f(s, \|\neg\top\|^{\mathcal{M}}_S) \cap \|\top\|^{\mathcal{M}}_S \neq \emptyset$. This then implies that $f(s, \|\neg\top\|^{\mathcal{M}}_S) \neq \emptyset$. However, from the discussion of $\neg BB(\alpha \wedge \neg\alpha)$ we already have a characterisation of seriality for incoherent propositions and so this adds nothing new.

There are two ways around this dilemma. First, we could declare it to be simply annoying, and deal only with the class of models in which $f(s, \|\alpha\|^{\mathcal{M}}_S) \neq \emptyset$ for every s and α. Second, we could look on this as being a genuine defect, and ask what might be done to correct things. The difficulty appears to be that the framework is too weak to state that at a situation a sentence (with the exception of incoherent sentences) does not receive true support. In logics of implicit belief (and also for truth at worlds) this is not a problem, since $\mathcal{M}, w \models B\alpha$ iff $\mathcal{M}, w \not\models \neg B\alpha$. Here however if $\mathcal{M}, s \models_T B\alpha$ then it is not necessarily true that $\mathcal{M}, s \not\models_T \neg B\alpha$. The problem appears intrinsic to the approach at hand, and applies to all the previous systems and variants discussed previously.

Before proceeding to the second alternative, it is worth noting that if we consider the class of models in which $f(s, \|\alpha\|^{\mathcal{M}}_S) \neq \emptyset$ for all s and α, the resultant logic can be described by the following axiomatisation; a proof is given in the appendix.

Axioms:

1. $B(\alpha \wedge \beta) \supset B\alpha \wedge B\beta$.
2. $B\alpha \supset B(\alpha \vee \beta)$.
   $B\beta \supset B(\alpha \vee \beta)$.
3. $B\alpha \equiv B\neg\neg\alpha$.
4. $B(\alpha \wedge (\beta \vee \gamma)) \supset B((\alpha \wedge \beta) \vee \gamma)$.
5. $BB(\alpha \wedge \neg\alpha) \supset B\neg B(\alpha \vee \neg\alpha)$

Rules of Inference:

1. From $\vdash B\alpha \supset B\beta$ and $\vdash B\beta \supset B\gamma$ infer $\vdash B\alpha \supset B\gamma$.
2. From $\vdash B\gamma \supset (B\alpha \wedge B\beta)$ infer $\vdash B\gamma \supset B(\alpha \wedge \beta)$.
3. From $\vdash (B\alpha \vee B\beta) \supset B\gamma$ infer $\vdash B(\alpha \vee \beta) \supset B\gamma$.
4. From $\vdash B\alpha \supset B\beta$ infer $\vdash B\neg\beta \supset B\neg\alpha$.
5. From $\vdash B\alpha \supset B\beta$ infer $\vdash BB\alpha \supset BB\beta$.

The soundness of the last axiom is easily demonstrated by noting that at situations, if

$$f(s, \|\alpha \wedge \neg\alpha\|^{\mathcal{M}}_S) \subseteq \|\alpha \wedge \neg\alpha\|^{\mathcal{M}}_S$$

then

$$f(s, \|\alpha \vee \neg\alpha\|^{\mathcal{M}}_S) \cap \|\alpha \wedge \neg\alpha\|^{\mathcal{M}}_S \neq \emptyset.$$

We also have the theorem

$$BB(\alpha \wedge \neg\alpha) \supset BB\alpha \wedge B\neg B\alpha.$$

These results may be justified on the grounds that, if an agent believes that it believes an inconsistency, then it believes pretty much anything to do with the inconsistency.

5 A MODAL OPERATOR FOR CONCEIVABILITY

I have argued that the only formally coherent notion of accessibility that meets the adequacy criteria of Section 3 is the approach in which

$$M, s \models_T B\alpha \text{ iff } f(s, \|\alpha\|^{\mathcal{M}}_S) \subseteq \|\alpha\|^{\mathcal{M}}_S \quad \text{and}$$
$$M, s \models_F B\alpha \text{ iff } f(s, \|\alpha\|^{\mathcal{M}}_S) \cap \|\neg\alpha\|^{\mathcal{M}}_S \neq \emptyset.$$

As noted, $B\alpha \wedge B\beta \supset B(\alpha \wedge \beta)$ is not valid in this system while, given a specificity condition, $B(\alpha \wedge \beta) \supset B\alpha \wedge B\beta$ is. However this approach has the problem that it is impossible to "prevent" a meta-belief such as belief in $B\alpha$, unless α is incoherent. If we consider logics of implicit belief, it can be seen that the difficulty arises from the fact that the modality $\neg L$ is used in two distinct ways.

1. Negated belief: The sentence $\neg L\alpha$ has the reading that "it is not the case that $L\alpha$ is believed".
2. Absence of belief: If $\neg L\alpha$ is true, then by consistency, $L\alpha$ cannot be true. So indirectly $\neg L\alpha$ expresses the "absence" of the belief $L\alpha$.

Since $\neg L\alpha$ is true iff $L\alpha$ is not true in logics of implicit belief, these senses are both expressed by the modality $\neg L$. However this doesn't work for explicit belief, since true support of $\neg B\alpha$ does not require that $B\alpha$ not have true support. Phrased another way, while we may be able to state that a belief always has true (or false) support in an explicit belief logic, within the given framework we cannot state that a belief lacks true (or false) support.

This does not mean that we are completely out of luck however. What we require is the definition of a distinct modal operator that will allow us to assert that

something is not held. In the case where we wish to assert that $B\alpha$ lacks any support, then, to introduce a term, we can assert that α is *inconceivable* at a given situation. The intent is to introduce a stronger notion than $\neg B\alpha$, which asserts only that α is not believed; rather what we try to capture is that in the set of accessible situations, α receives no support, in that there is no accessible situation supporting the truth of α.

Now, intuitively, α is conceivable (written $C\alpha$) just when, in the set of accessible situations, there is a situation supporting the truth of α. Conversely, α is inconceivable ($\neg C\alpha$) just when there is no accessible situation wherein α receives true support. We obtain the definitions:

$$M, s \models_T C\alpha \text{ iff } f(s, \|\alpha\|_S^{\mathcal{M}}) \cap \|\alpha\|_S^{\mathcal{M}} \neq \emptyset.$$
$$M, s \models_F C\alpha \text{ iff } f(s, \|\alpha\|_S^{\mathcal{M}}) \cap \|\alpha\|_S^{\mathcal{M}} = \emptyset.$$

What this does is effectively separate the two senses of the modality $\neg B$, the first given by the definition of support of falsity of B, and the second given in the above definition of C. In terms of modal logics in general, we have separated two senses of the notion of "possibility". The additional modality gives us the necessary granularity required to deal with degrees of incompatible beliefs.

Clearly if something is inconceivable then it ought not to be believed. Hence we would expect that $\neg C\alpha \supset \neg B\alpha$ is valid, or equivalently that $B\alpha \supset C\alpha$ is valid. This last formula is valid if $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$ implies that $f(s, \|\alpha\|_S^{\mathcal{M}}) \cap \|\alpha\|_S^{\mathcal{M}} \neq \emptyset$, or simply that $f(s, \|\alpha\|_S^{\mathcal{M}}) \neq \emptyset$.

So again we are back at the notion of seriality. However, this time we can deal with it. If we introduce a designated atomic sentence, $\top$, that has true support at every situation, then since $\|\alpha\|_S^{\mathcal{M}} \subseteq \|\top\|_S^{\mathcal{M}}$, we obtain $f(s, \|\top\|_S^{\mathcal{M}}) \subseteq f(s, \|\alpha\|_S^{\mathcal{M}})$. If $f(s, \|\top\|_S^{\mathcal{M}}) \neq \emptyset$ then $f(s, \|\alpha\|_S^{\mathcal{M}}) \neq \emptyset$ for every α. We can axiomatically express seriality simply by asserting that $C\top$. This means too that in the base semantics

$$C\top \supset (B\alpha \supset C\alpha)$$

is valid.

There is a further principle that characterises the logic of the C and B operators:

$$\text{From } B\alpha \supset B\beta \text{ infer } C\alpha \supset C\beta.$$

For the remainder of the section I will deal with the class of models in which $C\top$ is valid, and so $\neg C\alpha \supset \neg B\alpha$ is also.

To begin with, we can disallow incoherent beliefs by asserting that they are inconceivable, viz. $\neg C(\alpha \wedge \neg\alpha)$. Hence, by definition of C,

$$M, s \models_T \neg C(\alpha \wedge \neg\alpha) \quad \text{iff}$$
$$f(s, \|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}}) \cap \|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}} = \emptyset.$$

Consequently $\neg C(\alpha \wedge \neg\alpha)$ is valid in the class of models with no incoherent situations, and so $\neg B(\alpha \wedge \neg\alpha)$ is also valid in this class of models. But this also means that $\mathcal{M}, s \models_T B(\alpha \wedge \neg\alpha)$ can never hold (since $f(s, \|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}}$, together with $f(s, \|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}}) \neq \emptyset$, contradicts the previous relation). Thus we exclude incoherent beliefs while avoiding the difficulties discussed in the last section. Clearly also, $\{B\alpha, B\neg\alpha\}$ remains satisfiable.

Consider next where we wish to exclude incompatible beliefs and so have that $B\alpha \supset \neg B\neg\alpha$ is valid. We obtain that $B\alpha \supset \neg B\neg\alpha$ is valid in the class of models in which $f(s, \|\alpha\|_S^{\mathcal{M}}) \cap f(s, \|\neg\alpha\|_S^{\mathcal{M}}) \neq \emptyset$ (as would be expected).

Third, consider the class of models in which $\neg C\alpha \supset B\neg\alpha$ is valid. This sentence expresses a weakened law of the excluded middle: if α is inconceivable, then it must be the case that its negation is believed. This sentence is not valid in general since, we can have $\neg C(\alpha \wedge \neg\alpha)$ be valid in a class of models (where there are no incoherent situations), but $B(\alpha \vee \neg\alpha)$ does not have true support at a situation (since neither α nor $\neg\alpha$ have true support at that situation). It is valid however (and not unexpectedly) in the class of models in which $f(s, \|\alpha\|_S^{\mathcal{M}}) = f(s, \|\neg\alpha\|_S^{\mathcal{M}})$ and in which at every situation at least the truth or falsity of every atomic sentence is supported. Since we are assuming seriality, $\neg C\alpha \supset C\neg\alpha$ is also valid, and so if a sentence is inconceivable, then its negation must be conceivable.

The relation $B\alpha \supset \neg C\neg\alpha$, describes an agent with a great deal of faith in its beliefs: if the agent believes α then it is inconceivable that $\neg\alpha$. This relation is valid in the class of models in which $f(s, \|\alpha\|_S^{\mathcal{M}}) = f(s, \|\neg\alpha\|_S^{\mathcal{M}})$ and in which at every situation at most the truth or falsity of every atomic sentence is supported. If we combine this relation with the previous, then we obtain that $\neg B\neg\alpha \equiv C\alpha$ and so the two aspects of modality $\neg B$ (captured by $\neg B$ and $\neg C$) collapse to a single notion of negated belief. We obtain that this sentence is valid in the class of models in which $f(s, \|\alpha\|_S^{\mathcal{M}}) = f(s, \|\neg\alpha\|_S^{\mathcal{M}})$ and in which at every situation exactly one of the truth or falsity of every atomic sentence is supported and so every situation is in fact a world.

6 CONCLUSION

In semantic models of implicit belief, the most common means of obtaining different logics is by altering the properties of an accessibility relation in a Kripke model. In this paper I have argued that the notion of an accessibility relation cannot be adopted wholesale into situation-based logics of explicit belief. Based on a number of presumably-uncontentious adequacy criteria, it is clear that no extant logic of explicit belief

can be fitted with an accessibility relation to yield an acceptable system, nor does it appear that an entirely adequate approach can be developed. The notion of an accessibility relation is generalised, and one of the possible generalisations is argued to be intuitively and formally plausible. However, this approach is also not entirely satisfactory; the difficulty, which is present in all such logics of explicit belief, is that the modality $\neg B$ conflates two distinct senses, which may be characterised as negated belief and absence of belief. An additional modal operator C is added to the logic to separate these senses; the intended reading of $C\alpha$ is "α is conceivable". It is shown that the aforementioned difficulties do not arise in the resultant system. Arguably then the approach provides a semantically-justifiable basis from which logics of explicit belief may be developed by means of this generalisation.

Acknowledgments

This research was supported in part by the Natural Science and Engineering Research Council of Canada grant A0884. The author is a member of the Institute for Robotics and Intelligent Systems (IRIS) and acknowledges the support of the Networks of Centres of Excellence Program of the Government of Canada, and the participation of PRECARN Associates Inc.

A Proof of Theorem

I deal with two classes of models: first, $\mathcal{M}_f =< \mathcal{S}, f, \mathcal{T}, \mathcal{F} >$, as in the body of the paper, and second, $\mathcal{M}_N =< \mathcal{S}, N_\mathcal{T}, N_\mathcal{F}, \mathcal{T}, \mathcal{F} >$, where $\mathcal{S}, \mathcal{T}, \mathcal{F}$ are as in the body of the paper, and $N_\mathcal{T}$ and $N_\mathcal{F}$ are functions from $\mathcal{S}$ to sets of subsets of $\mathcal{S}$. $N_\mathcal{T}$ specifies those explicit propositions believed by an agent at a situation while $N_\mathcal{F}$ specifies those beliefs whose falsity is not supported. $\mathcal{M}_N$ models are developed in [Del92], and soundness and completeness results are provided. Members of $\mathcal{M}_f$ and $\mathcal{M}_N$ will be called f-models and N-models respectively. The completeness result for f-models is shown by proving that there is a $1-1$ correspondence between f-models and a restriction of N-models, preserving support of truth and falsity. The restriction on N-models is necessary since N-models provide a finer-grained (although not by much) semantics than do f-models.

Definitions of truth are:

$$\mathcal{M}_f, s \models_\mathcal{T} B\alpha \text{ iff } f(s, \|\alpha\|_\mathcal{S}^\mathcal{M}) \subseteq \|\alpha\|_\mathcal{S}^\mathcal{M} \text{ and}$$
$$\mathcal{M}_f, s \models_\mathcal{F} B\alpha \text{ iff } f(s, \|\alpha\|_\mathcal{S}^\mathcal{M}) \cap \|\neg\alpha\|_\mathcal{S}^\mathcal{M} \neq \emptyset.$$

and

$$\mathcal{M}_N, s \models_\mathcal{T} B\alpha \text{ iff } \|\alpha\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{T}(s)$$
$$\mathcal{M}_N, s \models_\mathcal{F} B\alpha \text{ iff } \|\alpha\|_\mathcal{S}^\mathcal{M} \notin N_\mathcal{F}(s).$$

In addition we have the specificity constraints on $N_\mathcal{T}$ and $N_\mathcal{F}$:

If $\|\alpha\|_\mathcal{S}^\mathcal{M} \subseteq \|\beta\|_\mathcal{S}^\mathcal{M}$ then
if $\|\alpha\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{T}(s)$ then $\|\beta\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{T}(s)$ and
if $\|\alpha\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{F}(s)$ then $\|\beta\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{F}(s)$.

Definition A.1 An N-model is *extended* if it satisfies the conditions:

1. $\mathcal{S} \in N_\mathcal{T}(s)$.
   If $\|\alpha\|_\mathcal{S}^\mathcal{M} = \emptyset$ then $\|\neg\alpha\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{F}$.
2. (a) If $\|\alpha \wedge \neg\alpha\|_\mathcal{S}^\mathcal{M} = \emptyset$ then: if $\|\alpha\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{T}(s)$ then $\|\alpha\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{F}(s)$.
   (b) If $\|\alpha \vee \neg\alpha\|_\mathcal{S}^\mathcal{M} = \mathcal{S}$ then: if $\|\alpha\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{F}(s)$ then $\|\alpha\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{T}(s)$.
3. If $\|\alpha \wedge \neg\alpha\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{T}(s)$ then $\|\alpha \vee \neg\alpha\|_\mathcal{S}^\mathcal{M} \notin N_\mathcal{F}$.

The first condition is a non-triviality condition, corresponding to seriality in f-models. The second condition ensures that $N_\mathcal{T}$ and $N_\mathcal{F}$ coincide at worlds. The final condition captures the semantic import of the last axiom. Soundness and completeness of N-models is given in [Del92], so all we have to show is the correspondence of the models.

Theorem For every f-model there is a pointwise equivalent extended N-model, and vice versa.

Proof:

($\rightarrow$)

Let $\mathcal{M}_f$ be a f-model and define $N_\mathcal{T}$ and $N_\mathcal{F}$ by:

$$\|\alpha\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{T}(s) \text{ iff } f(s, \|\alpha\|_\mathcal{S}^\mathcal{M}) \subseteq \|\alpha\|_\mathcal{S}^\mathcal{M},$$
$$\|\alpha\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{F}(s) \text{ iff } f(s, \|\alpha\|_\mathcal{S}^\mathcal{M}) \cap \|\neg\alpha\|_\mathcal{S}^\mathcal{M} = \emptyset.$$

That $N_\mathcal{T}$ and $N_\mathcal{F}$ satisfy the specificity condition is shown as follows.

Assume $\|\alpha\|_\mathcal{S}^\mathcal{M} \subseteq \|\beta\|_\mathcal{S}^\mathcal{M}$; thus $f(s, \|\beta\|_\mathcal{S}^\mathcal{M}) \subseteq f(s, \|\alpha\|_\mathcal{S}^\mathcal{M})$. So if $\|\alpha\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{T}(s)$ then $f(s, \|\alpha\|_\mathcal{S}^\mathcal{M}) \subseteq \|\alpha\|_\mathcal{S}^\mathcal{M}$ (from the definition of $N_\mathcal{T}$); so $f(s, \|\beta\|_\mathcal{S}^\mathcal{M}) \subseteq \|\beta\|_\mathcal{S}^\mathcal{M}$ (by set containments); so $\|\beta\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{T}(s)$.
If $\|\alpha\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{F}(s)$ then $f(s, \|\alpha\|_\mathcal{S}^\mathcal{M}) \cap \|\neg\alpha\|_\mathcal{S}^\mathcal{M} = \emptyset$. But $\|\alpha\|_\mathcal{S}^\mathcal{M} \subseteq \|\beta\|_\mathcal{S}^\mathcal{M}$; so $\|\neg\beta\|_\mathcal{S}^\mathcal{M} \subseteq \|\neg\alpha\|_\mathcal{S}^\mathcal{M}$ (by a straightforward inductive proof); so $f(s, \|\beta\|_\mathcal{S}^\mathcal{M}) \cap \|\neg\beta\|_\mathcal{S}^\mathcal{M} = \emptyset$; hence $\|\beta\|_\mathcal{S}^\mathcal{M} \in N_\mathcal{F}(s)$.

The proof that $N_\mathcal{T}$ and $N_\mathcal{F}$ satisfy the conditions for an extended N-model is straightforward.

We need to also show that a situation supports the same sentences in each model. The proof is by induc-

tion on the complexity of a sentence; the only interesting case is for an explicit belief $B\alpha$.

1. $\mathcal{M}_f, s \models_T B\alpha$ iff $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$ iff $\|\alpha\|_S^{\mathcal{M}} \in N_T(s)$ (by definition of N_T) iff $\mathcal{M}_N, s \models_T B\alpha$.
2. $\mathcal{M}_f, s \models_F B\alpha$ iff $f(s, \|\alpha\|_S^{\mathcal{M}}) \cap \|\neg\alpha\|_S^{\mathcal{M}} \neq \emptyset$ iff $\|\alpha\|_S^{\mathcal{M}} \notin N_{\mathcal{F}}(s)$ (by definition of $N_{\mathcal{F}}$) iff $\mathcal{M}_N, s \models_F B\alpha$

($\leftarrow$)

This gets messy.

Let $\mathcal{M}_N$ be an N-model and define Δ_α and Θ_α by

$$\Delta_\alpha = \cap\{\|\gamma\|_S^{\mathcal{M}} \mid \|\gamma\|_S^{\mathcal{M}} \in N_T(s) \text{ and } \|\gamma\|_S^{\mathcal{M}} \subseteq \|\alpha\|_S^{\mathcal{M}}\},$$

$$\Theta_\alpha = \cap\{\mathcal{S} - \|\neg\gamma\|_S^{\mathcal{M}} \mid \|\gamma\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s) \text{ and } \|\gamma\|_S^{\mathcal{M}} \subseteq \|\alpha\|_S^{\mathcal{M}}\}.$$

Δ_α is the interstection of all propositions that are believed and that α subsumes; Θ_α is the interstection of the complement of the negation of all propositions that are not believed and that α subsumes. Define f by:

If $\|\alpha\|_S^{\mathcal{M}} \in N_T(s)$, $\|\alpha\|_S^{\mathcal{M}} \notin N_{\mathcal{F}}(s)$, then
$f(s, \|\alpha\|_S^{\mathcal{M}}) = \Delta_\alpha$.
If $\|\alpha\|_S^{\mathcal{M}} \notin N_T(s)$, $\|\alpha\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s)$, then
$f(s, \|\alpha\|_S^{\mathcal{M}}) = \Theta_\alpha$.
If $\|\alpha\|_S^{\mathcal{M}} \in N_T(s)$, $\|\alpha\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s)$, then
$f(s, \|\alpha\|_S^{\mathcal{M}}) = \Delta_\alpha \cap \Theta_\alpha$.
If $\|\alpha\|_S^{\mathcal{M}} \notin N_T(s)$, $\|\alpha\|_S^{\mathcal{M}} \notin N_{\mathcal{F}}(s)$, then
$f(s, \|\alpha\|_S^{\mathcal{M}}) = \mathcal{S}$.

A rationale for this definition is as follows:

If $\|\alpha\|_S^{\mathcal{M}} \in N_T(s)$ then we require an expression for f where $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$. For starters we might suggest that $f(s, \|\alpha\|_S^{\mathcal{M}}) = \|\alpha\|_S^{\mathcal{M}}$. However if $\|\gamma\|_S^{\mathcal{M}} \in N_T(s)$ and $\|\gamma\|_S^{\mathcal{M}} \subseteq \|\alpha\|_S^{\mathcal{M}}$ then we require also that $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq f(s, \|\gamma\|_S^{\mathcal{M}})$. This is ensured by the specification of Δ_α.
Analogously, if $\|\alpha\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s)$ then we want $f(s, \|\alpha\|_S^{\mathcal{M}}) \cap \|\neg\alpha\|_S^{\mathcal{M}} = \emptyset$. Thus, to begin with we could set $f(s, \|\alpha\|_S^{\mathcal{M}}) = \mathcal{S} - \|\neg\alpha\|_S^{\mathcal{M}}$. Again though if $\|\gamma\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s)$ and $\|\gamma\|_S^{\mathcal{M}} \subseteq \|\alpha\|_S^{\mathcal{M}}$ then we also want $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq f(s, \|\gamma\|_S^{\mathcal{M}})$. This is ensured by the specification of Θ_α.

We need to show that $\mathcal{M}_f$ satisfies the specificity condition: If $\|\alpha\|_S^{\mathcal{M}} \subseteq \|\beta\|_S^{\mathcal{M}}$ then $f(s, \|\beta\|_S^{\mathcal{M}}) \subseteq f(s, \|\alpha\|_S^{\mathcal{M}})$. But this follows easily from the observation that $\Delta_\beta \subseteq \Delta_\alpha$ and $\Theta_\beta \subseteq \Theta_\alpha$.

To finish, we need to also show that a situation supports the same sentences in each model. The proof again is by induction on the complexity of a sentence; again, the only interesting case is for an explicit belief $B\alpha$.

For support of truth we have:

1. $\mathcal{M}_N, s \models_T B\alpha$ iff $\|\alpha\|_S^{\mathcal{M}} \in N_T(s)$; hence $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$ (by the definition of f); so $\mathcal{M}_f, s \models_T B\alpha$.
2. Conversely assume that $\mathcal{M}_f, s \models_T B\alpha$ and so $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$. I will assume that $\|\alpha\|_S^{\mathcal{M}} \notin N_T(s)$ and show that this leads to a contradiction. The two cases correspond to $\|\alpha\|_S^{\mathcal{M}} \notin N_{\mathcal{F}}(s)$ and $\|\alpha\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s)$.
   (a) If $\|\alpha\|_S^{\mathcal{M}} \notin N_T(s)$, $\|\alpha\|_S^{\mathcal{M}} \notin N_{\mathcal{F}}(s)$ then $f(s, \|\alpha\|_S^{\mathcal{M}}) = \mathcal{S}$. Since $f(s, \|\alpha\|_S^{\mathcal{M}}) \subseteq \|\alpha\|_S^{\mathcal{M}}$ we get that $\|\alpha\|_S^{\mathcal{M}} = \mathcal{S}$. But this contradicts $\mathcal{S} \in N_T$.
   (b) If $\|\alpha\|_S^{\mathcal{M}} \notin N_T(s)$, $\|\alpha\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s)$ then $f(s, \|\alpha\|_S^{\mathcal{M}}) = \Theta_\alpha = \cap\{\mathcal{S} - \|\neg\gamma\|_S^{\mathcal{M}} \mid \|\gamma\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s) \text{ and } \|\gamma\|_S^{\mathcal{M}} \subseteq \|\alpha\|_S^{\mathcal{M}}\}$. Simplifying, we get that $\cap\{\mathcal{S} - \|\neg\gamma\|_S^{\mathcal{M}} \mid \|\gamma\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s) \text{ and } \|\gamma\|_S^{\mathcal{M}} \subseteq \|\alpha\|_S^{\mathcal{M}}\} \subseteq \|\alpha\|_S^{\mathcal{M}}$ or $\mathcal{S} - \cup\{\|\neg\gamma\|_S^{\mathcal{M}} \mid \|\gamma\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s) \text{ and } \|\gamma\|_S^{\mathcal{M}} \subseteq \|\alpha\|_S^{\mathcal{M}}\} \subseteq \|\alpha\|_S^{\mathcal{M}}$ or $\cup\{\|\neg\gamma\|_S^{\mathcal{M}} \mid \|\gamma\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s) \text{ and } \|\gamma\|_S^{\mathcal{M}} \subseteq \|\alpha\|_S^{\mathcal{M}}\} \cup \|\alpha\|_S^{\mathcal{M}} = \mathcal{S}$. If there is no γ where $\|\gamma\|_S^{\mathcal{M}} \subset \|\alpha\|_S^{\mathcal{M}}$ then $\|\neg\alpha\|_S^{\mathcal{M}} \cup \|\alpha\|_S^{\mathcal{M}} = \|\neg\alpha \vee \alpha\|_S^{\mathcal{M}} = \mathcal{S}$, contradicting the fact that $\mathcal{M}_N$ is extended (condition 2b).
   Otherwise consider a least $\|\gamma\|_S^{\mathcal{M}}$ satisfying $\|\gamma\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s)$ and $\|\gamma\|_S^{\mathcal{M}} \subseteq \|\alpha\|_S^{\mathcal{M}}$. We obtain that $\|\gamma\|_S^{\mathcal{M}} \notin N_T(s)$ (since otherwise we would have that $\|\alpha\|_S^{\mathcal{M}} \in N_T(s)$), and the same argument yields $\|\neg\gamma \vee \gamma\|_S^{\mathcal{M}} = \mathcal{S}$, again contradicting the fact that $\mathcal{M}_N$ is extended.

For support of falsity we have a similar argument.

1. $\mathcal{M}_N, s \not\models_F B\alpha$ iff $\|\alpha\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s)$; so $f(s, \|\alpha\|_S^{\mathcal{M}}) \cap \|\neg\alpha\|_S^{\mathcal{M}} = \emptyset$; thus $\mathcal{M}_f, s \not\models_F B\alpha$.
2. Conversely assume that $\mathcal{M}_f, s \not\models_F B\alpha$; so $f(s, \|\alpha\|_S^{\mathcal{M}}) \cap \|\neg\alpha\|_S^{\mathcal{M}} = \emptyset$. I will assume that $\|\alpha\|_S^{\mathcal{M}} \notin N_{\mathcal{F}}(s)$ and show that this leads to a contradiction. Again, there are two cases.
   (a) If $\|\alpha\|_S^{\mathcal{M}} \notin N_T(s)$, $\|\alpha\|_S^{\mathcal{M}} \notin N_{\mathcal{F}}(s)$ then $f(s, \|\alpha\|_S^{\mathcal{M}}) = \mathcal{S}$. Since $f(s, \|\alpha\|_S^{\mathcal{M}}) \cap \|\neg\alpha\|_S^{\mathcal{M}} = \emptyset$ we get that $\|\neg\alpha\|_S^{\mathcal{M}} = \emptyset$, and so $\|\alpha\|_S^{\mathcal{M}} \in N_{\mathcal{F}}(s)$ contradicting the fact that $\mathcal{M}_N$ is an extended model.
   (b) If $\|\alpha\|_S^{\mathcal{M}} \in N_T(s)$, $\|\alpha\|_S^{\mathcal{M}} \notin N_{\mathcal{F}}(s)$ then $f(s, \|\alpha\|_S^{\mathcal{M}}) = \Delta_\alpha = \cap\{\|\gamma\|_S^{\mathcal{M}} \mid \|\gamma\|_S^{\mathcal{M}} \in N_T(s) \text{ and } \|\gamma\|_S^{\mathcal{M}} \subseteq \|\alpha\|_S^{\mathcal{M}}\}$. So $\cap\{\|\gamma\|_S^{\mathcal{M}} \mid \|\gamma\|_S^{\mathcal{M}} \in N_T(s) \text{ and } \|\gamma\|_S^{\mathcal{M}} \subseteq \|\alpha\|_S^{\mathcal{M}}\} \cap \|\neg\alpha\|_S^{\mathcal{M}} = \emptyset$. If there is no γ where $\|\gamma\|_S^{\mathcal{M}} \subset \|\alpha\|_S^{\mathcal{M}}$ then $\|\alpha\|_S^{\mathcal{M}} \cap \|\neg\alpha\|_S^{\mathcal{M}} = \|\alpha \wedge \neg\alpha\|_S^{\mathcal{M}} =$

$\emptyset$, contradicting the fact that $\mathcal{M}_N$ is extended. Otherwise consider a least $\|\gamma\|_S^{\mathcal{M}}$ satisfying $\|\gamma\|_S^{\mathcal{M}} \in N_{\mathcal{T}}(s)$ and $\|\gamma\|_S^{\mathcal{M}} \subseteq \|\alpha\|_S^{\mathcal{M}}$. We obtain also that $\|\gamma\|_S^{\mathcal{M}} \notin N_{\mathcal{F}}(s)$ (since otherwise we would have that $\|\alpha\|_S^{\mathcal{M}} \notin N_{\mathcal{F}}(s)$), and the same argument yields $\|\gamma \wedge \neg\gamma\|_S^{\mathcal{M}} = \emptyset$, again contradicting the fact that $\mathcal{M}_N$ is extended.

References

[BP83] J. Barwise and J Perry. *Situations and Attitudes*. Bradford Books. MIT Press, 1983.

[Che80] B.F. Chellas. *Modal Logic*. Cambridge University Press, 1980.

[Del92] J.P. Delgrande. A framework for logics of explicit and implicit belief. *to appear*, 1992.

[Ebe74] R.A. Eberle. A logic of believing, knowing, and inferring. *Synthese*, 26:356–382, 1974.

[FH85] R. Fagin and J.Y. Halpern. Belief, awareness, and limited reasoning. In *Proc. IJCAI-85*, Los Angeles, Ca, 1985.

[Haa86] A.R. Haas. A syntactic theory of belief and action. *Artificial Intelligence*, 28:245–292, 1986.

[Hin62] J. Hintikka. *Knowledge and Belief: An Introduction to the Logic of the two Notions*. Cornell University Press, 1962.

[HM92] J.Y. Halpern and Y.O. Moses. A guide to the completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992.

[Kon84] K. Konolige. A deduction model of belief and its logics. Technical report, Stanford, CA, 1984.

[Lak86] G. Lakemeyer. Steps towards a first-order logic of explicit and implicit belief. In *Proc. Theoretical Aspects of Reasoning About Knowledge Conference*, pages 325–340, Monterey, Ca., 1986.

[Lak87] G. Lakemeyer. Tractable meta-reasoning in propositional logics of belief. In *Proc. IJCAI-87*, pages 402–408, Milano, 1987.

[Lev84a] H.J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212, 1984.

[Lev84b] H.J. Levesque. A logic of implicit and explicit belief. In *Proc. AAAI-84*, Austin, TX, 1984.

[Lev85] H.J. Levesque. Appendix to: A logic of implicit and explicit belief. Unpublished manuscript, 1985.

[LL88] G. Lakemeyer and H.J. Levesque. A tractable knowledge representation service with full introspection. In *Proc. Second Theoretical Aspects of Reasoning About Knowledge Conference*, pages 145–159, Monterey, Ca., 1988.

[McA88] G. McArthur. Reasoning about knowledge and belief: A survey. *Computational Intelligence*, 4(3):223–243, Aug. 1988.

[MH79] R.C. Moore and G. Hendrix. Computational models of beliefs and the semantics of belief-sentences. Technical Report 187, SRI International, Menlo Park, 1979.

[Moo80] R.C. Moore. Reasoning about knowledge and action. Technical Report 181, SRI International, 1980.

[Sta87] R.F. Stalnaker. *Inquiry*. Bradford Books. MIT Press, 1987.

A STUDY IN THE LOGIC OF INTENTION

M.D. SADEK

Centre National d'Etudes des Télécommunications
LAA/TSS/RCP - B.P.40 22301 Lannion Cedex FRANCE
Email: sadek@lannion.cnet.fr

Abstract

We present a theory of intention which can be embedded in the model of an autonomous agent. Keeping Cohen and Levesque's global methodological framework (Cohen & Levesque 1986, 1990) we highlight problems which arise in their theory and provide solutions. In our approach, we adopt a *subjective point of view* (in order that an agent be given autoepistemic abilities (Levesque 1990) and, consequently, an own "existence") and we provide original notions such as *explicit introspection*, *relevant choice*, and *intention in a multiagent context* (which is different from the notion of joint intention introduced in (Levesque et al 1990). Moreover, in order to handle intention from a computational standpoint, we introduce and formalize a new concept: *need* or *potential intention*, which bridges the gap between a declarative and an operational semantics of intention.

1 INTRODUCTION

The concept of intention is of primary interest for autonomous agent modelling. It has been used in many works in AI (e.g., (Allen & Perrault 1980), (Appelt 1985), (Cohen & Perrault 1985), (Perrault & Allen 1980), but generally with an intuitive semantics or with a specification which is merely a placeholder for a "real" theory of intention. Recently, Cohen and Levesque (1986, 1990) (C&L hereafter) provided a logical analysis of this concept (referring mainly to Bratman (1984, 1987)'s work, for the philosophical aspects). They analyzed intention as a composite concept based on belief, choice and commitment. The agent's choices characterize the way the agent "desires" the *current* world to be (at the present moment).

C&L carried out their analysis as follows: first of all, *achievement goal* is defined: an agent has an achievement goal whenever she chooses the fact that a property which is believed to be false will become true. (In the following, the term "property" will be used to mean "state of affairs".) But an achievement goal does not constrain the agent to act in order to reach the future she has selected, nor even to maintain (for a sufficient time) her choice. Thus, the goal must be a *persistent goal* i.e., a goal that the agent will not abandon until she believes it has been satisfied or she believes it cannot be reached. But, an agent may indefinitely maintain her goal without ever attempting to achieve it. C&L define *intention* as a persistent goal which urges the agent to take action.

The methodological framework provided by C&L is of great interest for analyzing the concept of intention and for formalizing theories about it. However, in our opinion, problems of several kinds arise at various levels of their theory.

One of the problems concerns the logical characterization of the achievement goal. According to this characterization, whenever an agent believes that a property is false yet possible, she is considered as having this property as an achievement goal. In our opinion, this is due to an insufficient specification of the (type of) choice characterizing the achievement goal. In more general terms, there is a lack of a typology for the concept of choice.

Another problem concerns the definition of the intention to bring about a state of affairs. Two aspects of this definition are debatable. First, in our opinion, it deals with a nontypical case of intention, namely where the agent is primarily concerned with achieving the result of her intention by *herself*, and not with the result itself; thus, the fact that this result already holds does not prevent the agent from intending to achieve it *again*. Second, the definition of intention constrains the agent to be the *unique* "performer" of the events leading to her goal, i.e., it does not enable the agent to plan in such a way that actions from other agents can be part of her plan.

In other respects, C&L's approach does not account for a subjective point of view and is often of an *a posteriori* nature. For example, the characterization of certain of the agent's behaviors referring to "a slice" of her evolution (such as persistence) does not express the agent's point of view but that of an external observer examining the agent's evolution. With this approach, one cannot say whether, at a given moment, an agent actually has (respectively, has not) some given mental attitude; one can only say whether the agent has behaved as if she has (respectively, has not) some given mental attitude. Hence, the resulting theory does not necessarily characterize introspective agents; so, it cannot be straight forwardly embedded in the operational model of an *autonomous* agent (and does not allow for an autoepistemic viewpoint).

In this paper, we develop a formal theory of intention which does not have the problems mentioned, and we extend it to capture a computational view of intention.

2 OUTLINE

After having defined the formalism and its semantical model, we provide a logical analysis of the concept of choice and propose a typology of this concept. Then, we present (in our formalism) C&L's characterizations for goal, persistence and intention. Whenever necessary, we highlight the problems which their characterizations entail and we propose solutions for them. As the theory is being built, we provide original valid properties which enable us to have an insight into the logical behavior of each of the concepts introduced. Then, we examine the relevance of the intention theory to the commonsense world. Finally, we show that, from the computational standpoint, with the strict definition of intention, certain computation processes of an agent cannot be justified. To overcome this limit, we define, within the logic, a new concept: *need* or *potential intention*. This variant of intention bridges the gap between a declarative interpretation and a computational interpretation of intention, and is involved in any logical specification of a problem solving process and, thus, of a planning process.

3 BASES OF THE FORMAL MODEL

3.1 SYNTAX

The logic developed here is formalized in a first order modal language with equality, termed L. In addition to individual objects and agents, the universe of discourse involves sequences of (types of) events. (For our concern here, as in C&L's model, the same event can take place in different worlds; thus, events may be thought as event types. Moreover, only events performed by an agent are considered. We also assume that there are no simultaneous primitive events.) To allow for denoting primitive events, the language involves terms called event descriptors. For simplicity (and in the same manner as C&L), we consider a logic with no singular terms except the constant *NIL* which denotes the void event (type). To talk about complex "plans", action expressions are defined (using, in part, dynamic logic operators (see, e.g., (Harel 1984) or (Rosenchein 1981)) in the following way:

- event descriptors are action expressions;
- if a_1 and a_2 are action expressions so are "$a_1;a_2$", "$a_1 \mid a_2$" and "a_1^*" ; they respectively denote sequence action, nondeterministic choice action and iterative action.
- if i denotes an agent and ϕ a (closed) well formed formula (wff) then $(i, \phi?)$ is an action expression which we call *introspection-action*;

The set of variable symbols involves symbols which range over the set of event sequences; $e, e', \ldots e_1, e_2, \ldots$. Note that the quantification domain cannot include the action expressions in general. So, as with the formula schemas, which will be noted ϕ or ψ, the language includes schematic variables $a, a', \ldots a_1, a_2, \ldots$ which can be instantiated with any action expression. The variable symbols i and j range over the set of agents; for conciseness, when they have to be universally bound, these variables are left free and are intended to be schematic.

The (semantically primitive) operators $B, C, EB, Feasible, Done, Starts$ and $Agent$ are introduced. In addition to the wffs built in the usual way with predicate symbols, equality symbol, terms and the usual connectives and quantifiers, new wffs are obtained as follows: if a denotes an action expression, e and e' sequences of events and i an agent, and if ϕ is a wff, $B(i,\phi)$, $C(i,\phi)$, $EB(i,\phi)$, $Feasible(a,\phi)$, $Done(a,\phi)$, $Starts(e',e)$ and $Agent(i,e)$ are wffs. They are respectively understood to mean that ϕ follows from i's beliefs, ϕ follows from i's choices, i *explicitly* believes ϕ, a can take place and, if it does, ϕ will be true after that, a has just taken place and ϕ was true before that, e' denotes an initial subsequence of the sequence denoted by e and, finally, i denotes the *only* agent of the (sequence of) event(s) denoted by e.

3.2 FORMAL SEMANTICS

The language is interpreted in a possible-worlds based semantical structure. As is usually the case for belief, choice and events are characterized by accessibility relations between possible worlds. A model structure is a ten-uple $(\mathcal{W}, \mathcal{Agt}, \mathcal{B}, \mathcal{C}, \mathcal{E}, \mathcal{Evt}, \mathcal{Obj}, \mathcal{AGT}, \mathcal{ATT}, \sigma)$: $\mathcal{W}$ "is" a set of possible worlds; $\mathcal{B}$ and $\mathcal{C}$ are sets of binary relations on $\mathcal{W}$, both of them are isomorphic to the set of agents $\mathcal{Agt}$; $\mathcal{E}$ is a set of binary relations on $\mathcal{W}$, isomorphic to the set of primitive (types of) events $\mathcal{Evt}$; $\mathcal{Obj}$ is a set of objects; $\mathcal{AGT}$ is a function from $\mathcal{Evt}^+$ to $\mathcal{Agt}$; $\mathcal{ATT}$ is a set of functions from $\mathcal{W} \times \mathcal{Agt}$ to the powerset of the set of wffs, is isomorphic to $\mathcal{Agt}$; $\mathcal{ATT}$ is intended to characterize the attentional states of the agents as in

(Fagin & Halpern 1987); finally, σ is a set of first order truth assignments, and is isomorphic to $\mathcal{W}$. Before defining the satisfiability relation, let us see how the accessibility relations for action expressions, between two worlds w and w' of $\mathcal{W}$, are built using an assignment V. Given an agent i, a variable assignment v, and action expressions a_1 and a_2, we have:

- $w\mathcal{E}_{V(NIL)}w'$ iff w' is w ;
- $w\mathcal{E}_{V(e)}w'$ iff $w\mathcal{E}_{v(e)}w'$, if e denotes a primitive event (type) ;
- $w\mathcal{E}_{V(a1;a2)}w'$ iff $w\mathcal{E}_{V(a1)}w_1$ and $w_1\mathcal{E}_{V(a2)}w'$, for some w_1 ;
- $w\mathcal{E}_{V(a_1|a_2)}w'$ iff $w\mathcal{E}_{V(a_1)}w'$ or $w\mathcal{E}_{V(a_2)}w'$,
- $w\mathcal{E}_{V((i,\phi?))}w'$ only if $w \models EB(i,\phi)$ and w' is identical to w as far as implicit beliefs about primitive propositions are concerned.

The last assertion, which is a partial definition, can be explained in two steps: (1) An agent can ("physically") introspect about a property only if she is attentive to it. (2) The event denoted by an introspection-action may result in a world which is different from the initial one. For example, if attention is interpreted as the computational ability to answer a question in a limited time and/or space, after an introspection-action, an artificial agent may have consumed some of its resources and be in a logical configuration, different from the initial one, as far as her attentional state is concerned.

Now, given a model M, a possible world w of M and a variable assignment v, the satisfiability relation, which we write $\models$, is defined as follows: (For conciseness, we omit the part dealing with the constructs of first-order logic with equality:

- $(M,w,v) \models B(i,\phi)$ iff for all w' such that $w\mathcal{B}_{v(i)}w'$, $(M,w',v) \models \phi$;
- $(M,w,v) \models C(i,\phi)$ iff for all w' such that $wC_{v(i)}w'$, $(M,w',v) \models \phi$;
- $(M,w,v) \models EB(i,\phi)$ iff $(M,w,v) \models B(i,\phi)$ and $\phi \in \mathcal{RTI}(w,v(i))$;
- $(M,w,v) \models Feasible(a,\phi)$ iff for some w', $w\mathcal{E}_{V(a)}w'$ and $(M,w',v) \models \phi$;
- $(M,w,v) \models Done(a,\phi)$ iff for some w', $w'\mathcal{E}_{V(a)}w$ and $(M,w',v) \models \phi$;
- $(M,w,v) \models Agent(i,e)$ iff $v(i)$ is identical to $\mathcal{AGT}(v(e))$;
- $(M,w,v) \models Starts(e',e)$ iff $v(e')$ is an initial subsequence of $v(e)$.

A wff ϕ is *satisfiable* if for some (M,w,v), $(M,w,v) \models \phi$. A wff is *valid* if for every (M,w,v), $(M,w,v) \models \phi$; this will be written $\models \phi$.

3.3 CONSTRAINTS AND PROPERTIES

The relations in B are taken to be serial, transitive and euclidian and those in C are (initially) taken to be serial. Therefore, for a given agent, the logical model for belief is a *KD45*-model (with belief necessitation) and the one for choice is (for the moment) a *KD*-model (with choice necessitation) (see, e.g., Halpern & Moses 1985): both beliefs and choices are closed under logical consequence and are consistent; moreover, agents are positively and negatively introspective. In other respects, for all the agents, the property $C_{v(i)} \subseteq \mathcal{B}_{v(i)}$ termed (by C&L) the *realism constraint*, is imposed on the model, i denoting an agent. Hence:

$$\models B(i,\phi) \Rightarrow C(i,\phi)$$

The following abbreviations, introducing temporal modalities, are needed:

- $Possible(\phi) \stackrel{def}{\equiv} (\exists e)Feasible(e,\phi)$
- $Henceforth(\phi) \stackrel{def}{\equiv} \neg Possible(\neg\phi)$
- $Before(\phi,\psi) \stackrel{def}{\equiv} (\forall e)(Feasible(e,\psi) \Rightarrow (\exists e_1)starts(e_1,e) \wedge Feasible(e_1,\phi))$

Below, as in C&L, we will term the valid properties of the model "**Propositions**", and the formal constraints that filter only the semantical models which make valid these constraints valid "**Assumptions**"; we assume that there is at least one such model. As regards the proposition proofs, only guide-lines will be provided. In particular, the use of the belief sytem (*quantified-*) *KD45* will not be explicitly mentioned. (For the detail of the proofs, see (Sadek 1991a).)

Two useful properties:

Proposition 1: $\models (\forall e)\phi(e) \Rightarrow (\forall e_1)(\forall e_2)\phi(e_1;e_2)$

where e, e_1 and e_2 are variables ranging over the set of event sequences, and ϕ denotes any property.

This proposition follows from the definition of an event sequence. It means that if a property is true for all the event sequences, then, for each event sequence, it is true for all its initial event subsequences.

Proposition 2:

(i) $\models Before(\phi,\psi) \wedge henceforth(\phi \Rightarrow \phi') \Rightarrow Before(\phi',\psi)$

(ii) $\models Before(\phi,\psi) \wedge henceforth(\psi' \Rightarrow \psi) \Rightarrow Before(\phi,\psi')$

This proposition directly follows from the definitions of the operators *Before* and *Feasible* (using the consequential closure of the operator *Feasible*).

Action-introspection and belief:

Proposition 3: (i) $\models Feasible((i,\phi?),True) \Leftrightarrow EB(i,\phi)$

(ii) $\models Done((i,\phi?),True) \Rightarrow B(i,\phi)$

This proposition directly follows from the defintions of the operators *Feasible*, *Done*, *B* and *EB* (*True* being the propositional constant interpreted as usual). Note that $Done((i,\phi?),True)$ does not imply $EB(i,\phi)$.

3.4 COMPARISON WITH C&L'S MODEL

In C&L's model, there is no notion of introspection-action. However, there is the notion of test-action, defined as follows: if ϕ is a wff, $\phi?$ is an action expression. It is introduced in order to characterize what holds before or after an event, and then to impose on a course of events to satisfy a given state of affairs. Introspection-action denotes an event resulting from the performance of a particular action. Clearly, this kind of action is essential when specifying an artificial agent, because it directly refers to the computational introspection ability of the agent.

In other respects, there is a basic difference between C&L's model and ours, even if both are based on the possible worlds concept. In C&L's model, a possible world consists of a sequence of events, temporally extended infinitely in past and future. A formula is interpreted not only relatively to a world but also to a temporal index in this world. When an index in a world is given, so is an *inevitable* future. The C&L's formula *(HAPPENS e)* means that the sequence of events *e* is (immediately) inevitable. (C&L read it "*e* happens next".)

The model we adopt is similar, on the one hand, to a branching-time dynamic-logic model (Harel 1984) (Rosenchein 1981), and, on the other hand, to Moore's model for a unified logic of knowledge and action (Moore 1980). (In our model, a possible world can be considered of as a C&L's index in a course of events.) Thus, selecting a possible world does not impose a unique future. The satisfiability relation can thus only refer to the *possibility* of some sequence of events and not to its inevitability. This is the case for the operator *Feasible*.

In our opinion, C&L's model leads to a kind of "semantical determinism" when predicting evolution. C&L argue that their choice is motivated by the model ability to support beliefs about what was in fact about to happen. Strictly speaking, C&L's model is able to support beliefs about what is inevitable. Their purpose is achieved at the expense of a stronger constraint and to the detriment of the ability to support beliefs about what is possible. In addition, the conditions under which what is about to happen has the same meaning as what is inevitable, seemingly remain to be determined.

Rao & Georgeff (1991) also adopted a branching-time-based possible-worlds semantics. However, their approach differs from ours in that in their approach, a possible world is intrinsically a branching-time structure, while in ours, the branching-time structure is engendered by the existence of event accessibilty relations between possible worlds, and a possible world only corresponds to an index point in this structure.

4 STARTING THE THEORY

Future and autoreflexivity:

The agents we are modelling are *autoreflexive*, i.e., from their point of view, the world is in keeping with their beliefs: the schema $B(i,(B(i,\phi)\Rightarrow\phi))$ is valid. This feature must be preserved in an agent's reasoning model about the future. For example, an agent who thinks that a property is not possible must also believe that she will never believe it. To formalize this behavior, we propose the following constraint:

Assumption 4:

$$\models B(i,(Feasible(a,B(i,\phi))\Rightarrow Feasible(a,\phi)))$$

Note that the validity of the constraint $Feasible(a,B(i,\phi))\Rightarrow Feasible(a,\phi))$ is not acceptable since the operator *B* is interpreted subjectively.

Corollary 5:

(i) $\models B(i,(\forall e)(Feasible(e,B(i,\phi))\Rightarrow Feasible(e,\phi)))$

(ii) $\models B(i,Possible(B(i,\phi))\Rightarrow Possible(\phi))$

(i) trivially follows from assumption 4. As regards (ii), it is derived from (i) using the valid schema $((\forall x)(\phi\Rightarrow\psi))\Rightarrow((\exists x)\phi\Rightarrow(\exists x)\psi)$ and the definition of the operator *Possible*. Note that the directions "<=" are not valid: an agent may believe that there is no sequence of events after which she will believe ϕ and still believe that ϕ is possible.

Is an agent responsible for her acts ?

We consider that an agent achieved an event *non accidentally* if, just before that, she (successfully) achieved the introspection-action concerning this event feasibility. C&L adopt a weaker characterization (which, in our opinion, is less realistic, especially from a computational viewpoint) for the non accidental achievement of an event: the feasibilty (i.e., according to them, the inevitability) of the event must be an implicit consequence of the agent's beliefs.

In general, an agent may achieve an event accidentally. However, if this possiblity must be ruled out (which sounds reasonable, for example in the case where the actions are only linguistic actions, or in the case where the modelled agent has no physical effectors), the following property must be imposed on the semantical model:

$$\models Done(a)\wedge Agent(i,a)\Rightarrow Done((i,Feasible(a)?);a)$$

In all the cases, an agent must know what she has just done whenever she knows that she has just done something. The following property accounts for this ability:

Assumption 6:

$$\models B(i,(\exists e)Done(e)\wedge Agent(i,e))\Rightarrow (\exists e)B(i,Done(e)\wedge Agent(i,e))$$

Note that this property enables an agent to be the agent of some agent, unbeknowst to her. In other respects, note that the direction "<=" is valid.

5 THE BASES FOR A THEORY OF CHOICE

5.1 WHAT IS AN AGENT ENTITLED TO CHOOSE ?

First of all, let us rule out a possible "misunderstanding" about the concept of choice. C&L call "goal" what we call "choice". Naming "goal" a concept which satisfies the realism constraint may be misleading. Indeed, the notion of goal is commonly a future-oriented notion while the realism constraint also concerns the present (i.e., the proposition ϕ in the realism-constraint schema is not necessarily of the form *Possible*(ψ) or *Henceforth*(ψ)). Rao & Georgeff (1991) proposed variants of C&L's theory, mainly based on modifications (e.g., weakening) of the realism constraint. However, it seems that the main criticism they directed at C&L's theory are based on a (mis)interpretation of C&L's concept of goal in identifying it with the concept of achievement goal.

In fact, it is the realism constraint which provides the accurate interpretation one should have of the concept of choice (or goal, according to C&L's terminology). The choices of an agent are "reasonable preferences" about the *current* state of the world (i.e., "preferences" as far as possible according to the constraints of realism and consistency). These preferences may (and, generally, will) concern properties referring to the future, e.g., *Possible(p)*.

Among the set of worlds she considers as possible, an agent selects the preferred subset. This subset characterizes the (current) world as the agent desires it to be. It consists of three parts: *realistic choices*, i.e., those due to the realism constraint; *free choices*, i.e., the properties the agent desires and about which she is ignorant. (An agent i is said to be *ignorant* about ϕ if $\neg B(i,\phi) \wedge \neg B(i,\neg\phi)$.) the properties the agent is indifferent to. The following proposition shows how the agent's choices are constrained:

Proposition 7: $\models C(i,\phi) \Rightarrow \neg B(i,\neg\phi)$

This proposition follows from the consistency of choices and from the realism constraint. It states that an agent i can choose ϕ only if she does not believe $\neg\phi$. (It also should be noted that $\models C(i,B(i,\phi)) \Rightarrow C(i,\phi)$.) In other respects, an agent's choices are in perfect adequacy with her mental state:

Proposition 8: (i) $\models C(i,B(i,\phi)) \Leftrightarrow B(i,\phi)$
(ii) $\models C(i,\neg B(i,\phi)) \Leftrightarrow \neg B(i,\phi)$
(iii) $\models C(i,B(i,\phi)) \Rightarrow C(i,\phi)$

(i) and (ii) follows from proposition 7 and the realism constraint. (iii) is derived from (i) thanks to the realism constraint also. Note that the direction "<=" in (iii) is not valid, i.e., the schema $C(i,\phi) \wedge \neg C(i,B(i,\phi))$ is satisfiable. This is normal since an agent who is ignorant about p cannot choose $B(i,p)$ (in virtue of (ii) and of the choice consistency), while she can (freely) choose p.

5.2 CHOICE AND INTROSPECTION

As for her beliefs, an agent must, on the one hand, be able to introspect about her choices (resp. "non-choices") and, on the other hand, adopt the right choices (resp. "non-choices") according to what she believes about herself on this subject. To formalize this ability, we propose the following constraint:

Assumption 9: (i) $\models C(i,\phi) \Leftrightarrow B(i,C(i,\phi))$
(ii) $\models \neg C(i,\phi) \Leftrightarrow B(i,\neg C(i,\phi))$

Jointly used with the realism constraint (and with the choice consistency), this property leads to the following proposition:

Proposition 10: (i) $\models C(i,\phi) \Leftrightarrow C(i,C(i,\phi))$
(ii) $\models \neg C(i,\phi) \Leftrightarrow C(i,\neg C(i,\phi))$

A literal reading of this proposition may lead to a wrong and non intuitive understanding. For example, considering the direction "=>" of the schema (i), it should be noted that an agent who chooses ϕ because she believes ϕ, does not *freely* choose to choose ϕ, since she cannot avoid choosing ϕ; she *realistically* chooses to choose ϕ. In case the agent *freely* chooses ϕ, she also *freely* chooses to choose ϕ. Now, concerning the directions "<=", they mean that choosing to choose (or to not choose) can be reduced to merely choosing (or not choosing). (From the formal point of view, these directions allow for the reduction of the number of nested choice operators.) With (the directions "=>" of) proposition 10, the concept of choice (like the concept of belief) turns out to be characterized by a *KD45* logic (with choice-necessitation inference rule).

5.3 THE RELEVANT CHOICE

A formula such as $C(i,\phi)$ does not indicate whether the choice in question is free or realistic. Some behavior characterizations require the specification of the choice type. Note that $B(i,\phi)$ specifies a realistic choice while $C(i,\phi) \wedge \neg B(i,\phi)$ specifies a free choice.

However, in certain situations, one is primarily interested not so much in the type of choice as in how "significant" this choice is for the agent. In other words, if the choice is realistic, the question is to know whether the agent would have made it if she had been in a situation in which she would not have been committed to this choice by the realism constraint. Precisely, if this question is answered positively, we call this choice a *relevant choice* and formally define it as follows:

Definition 11:

$$RC(i,\phi) \stackrel{def}{=} C(i,\phi \wedge (\neg B(i,\neg\phi) \Rightarrow C(i,\phi)))$$

An agent *relevantly chooses* ϕ if she (realistically or freely) chooses ϕ and considers that the mere right to make this choice is a sufficient condition for making it.

The definition of the relevant choice enables to point out the agent's choices which are actually involved in her evolution process. In particular, it enables one to distinguish her real preferences among the agent's realistic choices.

As far as the agent's mental attitudes are concerned, the relevant choice has the same properties as the "mere" choice: propositions 8, and 10 remain true if RC is substituted for C. Moreover, concerning relevant choice, assumption 9 becomes a proposition. This result expresses the fact that the agent's mental attitudes are "significant" for her evolving process.

5.4 CHOICE AND IGNORANCE

A rational agent must not desire ignorance (or uncertainty) as a finality per se. Recall that if she is ignorant about some property, the agent relevantly choose this attitude. However, an agent must not prefer futures where she "looses knowledge" without the aim of "acquiring other knowledge". Formally, an agent can desire ignorance if the schema following schema is satisfiable:

$$C(i, Possible(\neg B(i,\phi) \wedge \neg B(i,\neg\phi))) \wedge (B(i,\phi) \vee B(i,\neg\phi))$$

If we want to rule out the possibilty for an agent to desire ignorance whatever be her motivation, the following property should be valid (having $\models \phi \Rightarrow Possible(\phi)$):

$$C(i, Possible(\neg B(i,\phi) \wedge \neg B(i,\neg\phi)) \Rightarrow (\neg B(i,\phi) \wedge \neg B(i,\neg\phi))$$

This property is too strong in that it does not allow an agent to believe that it may exist a sequence of events after which she will become ignorant about a given property. Also, it prevents an agent from envisaging ignorance as a "transition state". The following weaker property avoid this limitations:

Assumption 12:

$$\models C(i, Possible(\neg K(i,\phi))) \wedge K(i,\phi) \Rightarrow C(i, (\forall e)[Feasible(e, \neg K(i,\phi)) \Rightarrow \Rightarrow (\exists e')Feasible(e;e', K(i,\neg\phi))])$$

5.5 CHOOSING AND CHOOSING TO KNOW

We consider that if an agent prefers the futures where some property holds, she necessarily prefers the futures where she believes this property. (This point may raise some philosophical discussions about the view an agent has as regards her limited or unlimited life, and about the relation between knowledge and life. They are out of the scope of this work.) Therefore, we propose the following constraint for the semantical model, which, in practice, prevents an artificial agent (e.g., a robot) from behaving like a kamikaze:

Assumption 13:

$$\models C(i, Possible(\phi)) \Rightarrow C(i, Possible(B(i,\phi)))$$

6 GOAL AND PERSISTENCE

6.1 ACHIEVEMENT GOAL

C&L define an achievement goal as follows:

$$AG(i,\phi) \stackrel{def}{\equiv} C(i, Possible(\phi) \wedge \neg\phi) \wedge B(i,\neg\phi)$$

This definition can be formally simplified since the following equivalence is valid:

$$(C(i, Possible(\phi) \wedge \neg\phi) \wedge B(i,\neg\phi)) \Leftrightarrow (C(i, Possible(\phi)) \wedge B(i,\neg\phi))$$

According to this definition, whenever an agent believes simultaneously $\neg\phi$ and $Possible(\phi)$, she is committed to have ϕ as an achievement goal, because of the realism constraint. Fundamentally, this problem is due to a lack in the characterization of the choice in question. This choice must be a relevant choice. So, we propose to define an achievement goal as follows:

Definition 14: $AG(i,\phi) \stackrel{def}{\equiv} RC(i, Possible(\phi)) \wedge B(i,\neg\phi)$

With this definition, one can see that the following schema is satisfiable:

$$B(i, \neg\phi \wedge Possible(\phi)) \wedge \neg AG(i,\phi)$$

6.2 LOGICAL PROPERTIES OF THE ACHIEVEMENT GOALS

An agent agrees with her achievement goals (and "non-achievement goals"):

Proposition 15: (i) $\models AG(i,\phi) \Leftrightarrow B(i, AG(i,\phi))$
(ii) $\models AG(i,\phi) \Leftrightarrow C(i, AG(i,\phi))$
(iii) $\models \neg AG(i,\phi) \Leftrightarrow B(i, \neg AG(i,\phi))$
(iv) $\models \neg AG(i,\phi) \Leftrightarrow C(i, \neg AG(i,\phi))$

This proposition follows from definition 14 and assumption 9. The proof of (iv) uses, in addition, the validity of the schema $C(i,\phi) \vee C(i,\psi) \Rightarrow C(i,\phi \vee \psi)$ and proposition 10.

An agent cannot have a given property as an achievement goal without also having the belief of this property as an achievement goal. This is expressed by the following proposition, due to corollary 5(ii) and assumption 13: (Note that the direction "<=" is not valid.)

Proposition 15: $\models AG(i,\phi) \Rightarrow AG(i, B(i,\phi))$

The set of an agent's achievement goals is consistent: it is easy to show that the schema $AG(i,\phi) \Rightarrow \neg AG(i,\neg\phi)$ is valid. It is not closed under the conjunction: both the schemas $AG(i,\phi \wedge \psi) \wedge AG(i,\phi) \wedge \neg AG(i,\psi)$ and $AG(i,\phi) \wedge AG(i,\psi) \wedge \neg AG(i,\phi \wedge \psi)$ are satisfiable.

Now, let us examine the problem of the (non) consequential closure of the achievement goal, called the *side-effect* problem. For the analysis to be relevant, the "right" circumstances must be assumed, that is, given $AG(i,\phi)$ and $B(i,\neg\psi)$, in which case does $AG(i,\psi)$ hold ? Unless they state valid properties, the cases which do not characterize a mental attitude of the agent are rejected; so is $\phi \Rightarrow \psi$. It can be shown that the set of an agent achievement goals is closed only under the belief of perpetual equivalence (i.e., $B(i,Henceforth(\phi \Leftrightarrow \psi))$) and, a fortiori, under valid equivalence (i.e., $\models \phi \Leftrightarrow \psi$). Note that the case $B(i,Henceforth(\phi \Rightarrow \psi))$ does not necessarily entail $AG(i,\psi)$ since the relevancy feature concerning the choice of ψ is not necessarily satisfied.

6.3 PERSISTENT GOAL

C&L define a persistent goal as follows:

$$PG(i,\phi) \stackrel{def}{=} AG(i,\phi) \wedge Before((B(i,\phi) \vee B(i,\neg Possible(\phi))), \neg C(i,Possible(\phi) \wedge \neg \phi))$$

First, note that, because keeping or giving up the choice of $\neg\phi$ depends on believing or not believing $\neg\phi$, the persistence condition (i.e., the second term of the conjunction) is equivalent to

$$Before((B(i,\phi) \vee B(i,\neg Possible(\phi))), \neg C(i,Possible(\phi)))$$

C&L's definition of the persistent goal is of an *a posteriori* kind: one can never decide whether an agent has or has not some persistent goal at a given moment; one can just say whether she had it or not. (Moreover, the persistence condition may have been fulfilled accidentally.) The reason for this is that this definition does not express the agent's viewpoint. So, it does not fit an autoepistemic perspective. It is easy to be convinced by noting that this definition does not allow for introspection.

The solution is simple. To be allowed to claim an achievement goal, an agent has to relevantly choose the possibility of some property which she believes to be false; to be allowed to claim a persistent goal, the agent must also *choose* to keep this attitude until she thinks that the desired property has been satisfied or until she thinks it will never be true. So, we propose the following definition for persistent goal:

Definition 17:

$$PG(i,\phi) \stackrel{def}{=} AG(i,\phi) \wedge C(i,Before((B(i,\phi) \vee B(i,\neg Possible(\phi))), \neg C(i,Possible(\phi) \wedge \neg \phi)))$$

Cohen & Levesque (1991) remedied the problem of the lack of introspection of persistent goals with the addition of a KNOW (defined as usual) enclosing the persistence condition. However, while this modification works for handling introspection, it does not seem to be relevant for characterizing the basic agent's mental attitude when adopting persistence. Indeed, an agent may know something about her own behavior without *originally* having an internal commitment towards this behavior. The only attitude that accounts for an internal commitment is the attitude of choice (or goal). In that case, the agent "comes to know" because she "chooses" (in virtue of proposition 5) and not the converse (as it might have been the case, due to the realism constraint). In other respects, by using a KNOW instead of a BELIEF, Cohen & Levesque (1991) still impose on the course of events to satisfy the persistence condition (independently from the agent's point of view), which unfortunately keeps an *a posteriori* character to their definition of persistent goal.

6.4 LOGICAL PROPERTIES OF THE PERSISTENT GOALS

An agent agrees with her persistent goals (and with her "non-persistent goals"): the proposition 15 still holds if *PG* is substituted for *AG*. This can be proved by starting from proposition 15 and definition 17, and using the same properties of choice than those used for the proof of proposition 15.

An agent cannot have a given property as a persistent goal without also having the belief of this property as a persistent goal. This is expressed by the following proposition, due to corrolary 5(ii) and the realism constraint (and using the sound inference rule *if* $\models \phi$ *then* $\models Henceforth(\phi)$): (Note that the direction "<=" is not valid.)

Proposition 10: $\models PG(i,\phi) \Rightarrow PG(i,B(i,\phi))$

The persistent goal is first of all an achievement goal; so it is easy to show that the set of persistent goals is consistent, i.e., that the schema $PG(i,\phi) \Rightarrow \neg PG(i,\neg\phi)$ is valid. In other respects, neither $PG(i,\phi \wedge \psi)$ implies $PG(i,\phi) \wedge PG(i,\psi)$, nor conversely. Finally, for the same reasons as for the achievement goal, the set of persistent goals of an agent is closed only under belief of the perpetual equivalence, and, a fortiori, under the valid equivalence.

Concerning the consequential closure, C&L carried on an analysis similar to that we made for achievement goals. Our conclusions are in most cases different from theirs. In addition, the justifications we provide, seem to be more "motivated" than C&L's ones. Indeed, first of all, C&L's do not assume the "right" circumstances , i.e., they do not consider that the condition $B(i,\psi)$ holds. They appeal to that condition only for analyzing the cases $Henceforth(B(i,(\phi \Rightarrow \psi)))$ and $\models \phi \Rightarrow \psi$. As regards these cases, they answer "yes" to the question "Does $PG(i,\psi)$ necessarily follow from $PG(i,\phi) \wedge B(i,\neg\psi)$ if $Henceforth(B(i,Henceforth(\phi \Rightarrow \psi)))$ holds or if $\models \phi \Rightarrow \psi$ holds ?". Unlike them, we answer "no" to this question, for the same reasons as for the achievement goal. As regards the case $B(i,Henceforth(\phi \Rightarrow \psi))$ C&L claim that $PG(i,\psi)$ does not hold. They argue that the agent's belief may change, and, so, that the agent need no longer choose worlds in which $\phi \Rightarrow \psi$ holds, thus needing no longer have ψ as a goal; the agent would have dropped

his "goal" for reasons other than those stipulated by the definition of persistent goal, and so does not have it as a persistent goal. We consider that this argument is inadmissible. The hypothesis saying that the agent's belief may change accounts for a viewpoint external for the agent; the agent *now* thinks that $\phi \Rightarrow \psi$ will *always* hold. The question is to know whether, starting from this point of view, the agent *now* has ψ as a persistent goal.

Before going further, we would like to point out, without going into detail here, the following problem (which we call *the commitment problem*): is an agent committed to her commitment, and if she is, how is this formally captured ? For our concern here, we suppose that a "first level" commitment is (necessary and) sufficient for characterizing the concept of persistent goal because, in our theory, it is a distinctive feature of this concept.

7 INTENTION

7.1 C&L'S DEFINITIONS

C&L provide two definitions for intention: the first one concerns intention to do actions, and the second, intention to bring about a world state. The problems we address here mainly concern the secon defintion. But, let us first examine the differences between these two definitions.

As regards the intention to do actions, C&L give the following definition, recast within our logical framework:

$$I_1(i,a) \stackrel{def}{=} PG(i, Done(a, B(i, Feasable(a)))) \wedge Agent(i,a)$$

There is a formal difference between C&L's original definition and this one, worth mentioning: the fact that the agent intending to do a be the agent of a does not appear in C&L's definition as a natural prerequisite to be able to pose such a definition; the agenthood of a is scattered within the persistent goal argument as an additional property the agent has to achieve. This sounds odd: the actions the agent intends to do, intrinsically specifies their agent(s). Thus, an agent cannot intend to achieve an event of which she is not the agent.

As regards intention to bring about a property, C&L define it as follows:

$$I(i,\phi) \stackrel{def}{=} PG(i, \exists e(Done(e, \alpha(i,e,\phi)) \wedge Agent(i,e)))$$

where $\alpha(i,\phi)$ is the following abbreviation:

$$\alpha(i,e,\phi) \equiv B(i, \exists e'(Feasible(e',\phi) \wedge Agent(i,e'))) \wedge \neg C(i, \neg Feasible(e,\phi))$$

The left part of the conjunction $\alpha(i,e,\phi)$ is intended to constrain the agent not to act accidentally; moreover, it expresses the fact that the agent does not know in advance the whole sequence of events which will enable her to achieve ϕ (by herself). The right part of the conjunction is intended to deal with the troublesome case which occurs when the agent achieves her goal accidentally while acting to achieve it "intentionally" (see (Searle 1983), (Bratman 1987), C&L).

7.2 DO ACTIONS VS ACHIEVE PROPERTIES

The definition of intention $I(i,\phi)$ concerns an agent who does not beforehand know the sequence of action she will perform in order to bring about (herself) a world state. This is different from the context in which the definition $I_1(i,a)$ has to be considered. This definition characterizes an agent who, this time, knows the actions she intends to do. Consequently, one must not expect any equivalence between $I_1(i,a)$ and $I(i, Done(a)) \wedge Agent(i,a)$.

The hypothesis according to which, *a priori*, an agent does not always beforehand know a completely specified plan which will allow her to reach her goal, is quite reasonable. Nevertheless, the notion of completely (or incompletely) specified plan would deserve to be examined. Two cases, non necessarily distinct, can be explored. (1) The agent selects an action expression which includes nondeterministic choice actions and/or conditional actions. In this case, the action sequence is, *a priori*, not beforehand known to the agent. (2) The agent selects an action sequence including partially instantiated and/or non primitive (if the agent plans hierarchically) "actions", which will be instantiated or expanded during the execution process only. In this case, could one say that, before doing such an action sequence, the agent does not know what she is going to do ? In other words, knowing such an action sequence, does it mean for an agent, knowing a completely specified plan. These questions do not seem to be trivially answered "yes". We will not go deeper in this discussion. However, it is worth underlining that the point here is to know to what extent, regarding to what is meant by knowing (or not knowing) an action sequence, the intention to achieve some property can be reduced to the intention to do an action expression. Let us therefore examine the case where an agent who has the intention to bring about some property, beforehand determines the action sequence she intends to do in order to achieve her goal. The definition of $I(i,\phi)$ turns out to be formulated as follows, noting that, by virtue of the realism constraint and the choice consistency, the schema $B(i, Feasible(e,\phi)) \wedge \neg C(i, \neg Feasible(e,\phi))$ is logically equivalent to $B(i, Feasable(e,\phi))$.

$$I'(i,\phi) \stackrel{def}{=} PG(i, (\exists e)\,[Done(e, B(i, Feasible(e,\phi))) \wedge Agent(i,e)])$$

With this definition, the equivalence between $I_1(i,a)$ and $I'(i, Done(a)) \wedge Agent(i,a)$ still must not be expected. Indeed, on the one hand, $I'(i, Done(a)) \wedge Agent(i,a)$ does not imply $I_1(i,a)$, since according to the definition of $I_1(i,a)$ an agent cannot have the intention to do a if she (even intentionally) did the begining of a - say $a1$ -: this

definition require from the agent to think that she can (or, in the C&L's sense, that she is about to) do a before starting doing it. However, there is no reason to think that if the agent can (or is about to) do a, the agent can (or is about to) do a.

In the other hand, $I_1(i,a)$ does not imply $I'(i,Done(a)) \wedge Agent(i,a)$, since an agent who intends to do an action expression a, has the persistent goal that a be done, before what, she thinks that a is feasible, without necessarily knowing the action sequence that will be done; note that this is not the case with the intention to bring about the fact that a be done: in this case, the agent knows the action sequence she should do after which a will be done. In fact, it can be shown that $I_1(i,a)$ implies $I'(i,Done(a)) \wedge Agent(i,a)$ only when the action expression a corresponds to an action sequence.

7.3 PROBLEMS WITH C&L'S DEFINITION OF INTENTION AND SOLUTIONS

Now, we concentrate on the problems which arise from C&L's definition of the intention to bring about a state of affairs.

(1) To characterize the *"non accidental"* performance of an event e by an agent i, C&L content themselves with the fact that "e happens next" follows from i's beliefs. But this fact may be an *implicit* belief, which is out of the "scope" of i's attention. In our opinion, an agent performs an action non accidentally if, just before achieving it, she introspects successfully regarding the fact that the action is presently feasible, that is, the agent must be attentive to this fact.

(2) A particular feature of the agents described by C&L is the ability to claim an intention in order to bring about ϕ without necessarily having ϕ as an achievement goal. Indeed, believing ϕ does not prevent the intention of bringing it about, *by oneself*. In the characterization privileged by C&L, the primary interest of an agent is not the end result of her intention but that she will achieve this result *by herself*. In our opinion, this behavior is not a *typically* commonsense one. To characterize the behavior of agents primarily interested in the end result of their intentions, C&L propose to use the formula $PG(i,\phi) \wedge I(i,\phi,PG(i,\phi))$ where $I(i,\phi,PG(i,\phi))$ is the intention to bring about ϕ but which can be given up if $PG(i,\phi)$ no longer holds. In our opinion, *this* is the typical case for defining intention. In this case an agent can claim an intention to bring about a state of affairs only if she has this state of affairs as an achievement goal. When defining intention with this view, the (particular) agents primarily concerned with achieving ϕ *by themselves* will have the intention of bringing about $(\exists e)(Done(e,Feasible(e,\phi) \wedge \neg \phi)) \wedge Agent(i,e))$.

(3) Finally, C&L's definition does not allow for planning in a multiagent context since it constrains the agent to be *solely* concerned with the achievement of her goal. In our opinion, the intention of an agent to bring about a property has to be defined in such a way that the property in question may result from a potentially multiagent sequence of events, that is, it should be possible that the actions of other agents be a part of the agent's "plan". Note that neither the notion of joint intention introduced in (Cohen & Levesque 1990b) nor the the definition of INTEND* in (Cohen & Levesque 1991) rule out the problem raised here: the first one is concerned with what it means for agents to have common intentions, and the second one deals with a partially specified sequence of events but keeps the uniqueness of the sequence agent.

7.4 COOPERATION AND INTENTION: REDEFINING INTENTION

Following the criticisms above, we propose to define the intention to bring about a state of affairs as follows:

Definition 19:

$$I(i,\phi) \stackrel{def}{\equiv} PG(i,\phi) \wedge C(i,Coop(i,\phi) \wedge Pers(i,\phi,Coop(i,\phi)))$$

where $Coop(\phi)$ and $Pers(Coop(\phi))$ are the following abbreviations:

$$Coop(i,\phi) \equiv (\forall e)[B(i,(\exists e')Feasible(e;e',\phi)) \wedge Agent(i,e)) \Rightarrow C(i,Possible(Done((i,Feasible(e)?);e)))]$$

$$Pers(i,\phi,Coop(\phi)) \equiv Before((B(i,\phi) \vee B(i,\neg Possible(\phi))), \neg C(i,Coop(\phi))))$$

This definition can be explained in two steps. First, the agent necessarily has ϕ as a persistent goal, that is, she believes $\neg \phi$, she relevantly prefers $Possible(\phi)$ and she commits herself (for a sufficient time) to this preference. Second, the agent adopts an *active* behavior appropriate to a potentially cooperative multiagent context and commits herself to this behavior as long as she keeps her achievement goal. (Note that the two commitments could have been put together. But for the sake of clearness, we have preferred highlighting the persistent goal.)

The behavior in question is formalized in the definition above by $Coop(i,\phi)$. This formula means that the agent will do a sequence of events *by herself* whenever she thinks that this sequence will lead her *towards* her goal. Moreover, in her "plan", the agent takes the potential contribution of other agents (represented by e' in the formula) into account. Indeed, $Coop(i,\phi)$ says that the agent considers that the events she performs, potentially start a multiagent process, the end result of which is her goal. The contribution of the other agents can be reduced to nothing in the case where the mere agent's acting is sufficient to reach her goal. Moreover, it is worth noticing that, since the agent engages herself to maintain this behavior (for a sufficient time), it is applied at each step of the agent's evolution which will lead her towards her

goal. (In the definition above, this means that the behavior recursively applies to the subsequences remaining to be (planned and) done.)

Note that, since the formula $Coop(i,\phi)$ includes a condition about the agent beliefs, it urges the agent to search for the sequences of events possibly leading her to her goal, i.e., to plan. This meets one of Bratman's functional roles played by intention (Bratman 1984, 1987): *Intention normally poses problems to the agent; the agent needs to determine a way to achieve them.*

Finally, there is a point worth mentioning: the universal quantification stated in $Coop(i,\phi)$ means that the agent chooses all the futures where she would have acted to achieve her goal. The agent may plan all the "interesting" sequences of events and eventually pick one of them to perform, or she may plan one "interesting" sequence and perform it. In both cases, the agent will eventually choose one sequence to be performed and, after having performed it, obviously she will drop the others since she is no longer concerned with them.

7.5 APPLICATION TO THE DIALOGUE CASE

Let us consider an agent i who intends to acquire some information and who believes that her interlocutor j holds this information. According to the definition above, i's intention engenders the following behavior. The agent i plans a sequence of events $e1;e2$ where $e1$ is a request from i to j and $e2$ the corresponding reaction of j. In virtue of $Coop(i,\phi)$, i achieves $e1$ (i.e., requests j to provide her with the desired information) since she believes that $e1$ starts a process resulting in her goal and that she is the agent of this event. After this has been carried out, i does not cancel her intention since her goal is neither reached nor impossible. Indeed, there is nothing that i believes she can do by herself to evolve towards her goal but, henceforth, she knows that there is a possible event (i.e., the reaction of j, which she has triggered off) which will normally achieve her goal. Therefore, she waits for the event to take place.

7.6 LOGICAL PROPERTIES OF INTENTION

An agent agrees with her intentions (and with her "non-intentions"): proposition 15 still holds if I is substituted for AG. It can be proven similarly to proposition 15 for persistent goal, using this proposition and the definition 12. In other respects, an agent cannot intend to bring about a given property without intending to believe it. This is expressed by the following proposition, due to proposition 1, corrolary 5 and proposition 2: (Note that the direction "<=" is not valid.)

Proposition 20: $\models I(i,\phi) \Rightarrow I(i,B(i,\phi))$

The logical properties concerning achievement goal and persistent goal (i.e., consistency, non transparency to the conjunction and the conditions for the consequential closure) also hold regarding intention, since intention is, first of all, a persistent goal.

8 RELEVANCE OF THE THEORY OF INTENTION TO COMMONSENSE

We now explore the way the logical results presented above account for our intuition concerning the concepts dealt with here. For this purpose, we examine the basic principles of the definition of achievement goal.

At a first glance, one may consider the definition of achievement goal too strong as it sets the belief of $\neg\phi$ as a necessary condition to have ϕ as an achievement goal. Let us consider an agent i ignorant about ϕ. According to the definition in question, even if i desires ϕ, she cannot claim that ϕ is one of her achievement goals, since she cannot claim that she believes $\neg\phi$. The question is "What about a definition which merely requires $\neg B(i,\phi)$ instead of $B(i,\neg\phi)$?". In other words, to be entitled to claim ϕ as an achievement goal, is it necessary to specifically believe $\neg\phi$ or is it sufficient to not believe ϕ (and, therefore, to be ignorant about ϕ) ?

It is normal that an agent who is ignorant about ϕ cannot have ϕ as an achievement goal since she cannot claim here desire to achieve a property which is possibly already satisfied. Therefore, one may say that no matter how important ϕ is for the agent, she cannot act since she cannot claim that she has the intention to achieve ϕ. In fact, the explanation for this "problem" can be found in what the agent is specifically seeking. The agent does not exactly intend to change the state of ϕ (since this state may be the one she desires) but she specifically intends to change *her belief about* ϕ. Indeed, the agent cannot declare believing $\neg\phi$ but she can declare believing *not believing* ϕ. The property the agent desires (or intends to achieve) is not ϕ but $B(i,\phi)$.

Let us take an example. Suppose an agent i who is at home and who desires that the external door be closed (i.e., $C(i,Possible(p))$ where $p \equiv$"The door is closed") but who ignores the real state of the door (closed or open). Recall that in virtue of assumption 13, i desires to believe that the door is closed (i.e., $C(i,Possible(B(i,p)))$). If the agent decides to achieve her choice and goes to the door, her actions cannot be justified by the intention to bring about the fact that the door be closed but by the intention to bring about the fact that she comes to believe that the door is closed. Note that if, when she arrives at the door, the agent observes that it is closed, she would have in any case achieved her goal since, henceforth, she knows that the door is closed.

In a dialogue context, let us consider the case where an agent i desires that her interlocutor j believes ϕ (i.e., $C(i,Possible(B(j,\phi)))$) and who is ignorant about j's beliefs about ϕ. So, the agent i does not believe that j does not believe ϕ (i.e., $\neg B(i, \neg B(j,\phi))$). Thus, i cannot act to achieve $B(j,\phi)$. However, she can act to achieve $B(i,B(j,\phi))$.

9 NEED OR POTENTIAL INTENTION: A COMPUTATIONAL POINT OF VIEW

The definition of intention specifies the agent's behavior engendered by intention but, conversely, no information is provided about the way intention is engendered by the evolution process of an agent. As far as the specification of an artificial agent is concerned, this means that the theory does not exhibit an "inferential mechanism" enabling the agent to provide itself with "intentions", that is, to evolve autonomously. Thus, "the loop is not looped".

This remark is clearly justified when adopting a computational point of view. Suppose that in the planning process engendered by an intention a backward control strategy is used, and suppose that an action is selected (because its effects meet the agent's goal). Afterwards, the "process" has to check as to whether the precondition ϕ of the selected action is satisfied, otherwise the intention to achieve it will be inferred. To check whether or not the precondition is satisfied means to carry out a computation process resulting in the answer to the question "Is ϕ believed ?" The problem is that there is no intention attitude which motivates such a process. (Note that it cannot be motivated even by the intention to know whether ϕ, since it is not the case that the agent does not believe ϕ.) In fact, while from the logical standpoint the transition between a "non-intention" and the corresponding intention cannot be observed, this is not the case from the computational standpoint.

To solve this problem, we propose to characterize the mental attitude which, in the agent evolution, engenders intention. Thus, we introduce the concept of *need* or *potential intention*. An agent's needs (to believe) ϕ if, in the case that ϕ is not believed, she adopts the intention to believe ϕ. In other words, an agent intends to believe ϕ if she needs ϕ and does not believe ϕ.

From a logical point of view, there is a *bilateral* inference relationship between need and intention: a non satisfied need engenders an intention and the achievement process triggered off by an intention engenders a need. Now, "the loop is looped". (To see how this relationship underlies a (speech act) planning process, see (Sadek 1990, 1991a, 1991b).)

The concept of need is a latent form of intention and, so, it can be logically characterized from (choice,) belief and intention. We formalize it by the modal operator N and propose to define it as follows: (in our previous works mentioning this concept, we note it W it order to relate it to the usual "Want" notion.)

Definition 21: $N(i,\phi) \stackrel{def}{=} C(i, \neg B(i,\phi) \Rightarrow I(i,B(i,\phi)))$

This definition calls for two comments. First, the potential intention is the (necessarily relevant) choice of a certain behavior. Second, the intention is about $B(i,\phi)$ and not about ϕ. According to this definition, an agent needs ϕ if the fact that she does not believe ϕ is a sufficient condition for intending to believe ϕ; it is not said that ϕ is not believed because $\neg\phi$ is believed. If this is the case, the intention of believing ϕ remains legitimate and, possibly, engenders the intention to bring about ϕ.

The definition of need is "designed" in such a way to allow for the two cases concerning the lack of a given property from an agent's beliefs. Finally, note that behind the implication which appear in the definition, an equivalence is hidden.

Logical properties of need:

An agent agrees with her needs (and with her "non-needs"): the proposition 15 still holds if N is substituted for AG. In other respects, if an agent needs ϕ (to hold), she needs to believe ϕ, and conversely (unlike intention). In virtue of definition 18 and of the validity of the schema $(I(i,\phi) \wedge I(i,\phi \Rightarrow \psi)) \Rightarrow I(i,\psi)$, we have the following proposition:

Proposition 22: $\models N(i,\phi) \Leftrightarrow N(i,B(i,\phi))$

If an agent needs ϕ and does not believe ϕ, she intends to believe ϕ and conversely.

Proposition 23: $\models (N(i,\phi) \wedge \neg B(i,\phi)) \Leftrightarrow I(i,B(i,\phi))$

This proposition comes from proposition 8(ii) and from $\models I(i,\phi) \Leftrightarrow C(i,I(i,\phi))$ (proposition 15 for intention).

Note the computational interest of the inference which follows from this proposition: if $N(i,\phi) \wedge \neg B(i,\phi)$ then infer $I(i,B(i,\phi))$. In our opinion, it represents the basis of any logical specification of a problem solving process and, therefore, of a planning process.

Unlike intention, an agent may need a property and, at the same time, believe it. (Also, an agent who believes a property, of course, does not necessarily need it.) It can be easily proved that the following schemas are satisfiable: $N(i,\phi) \wedge B(i,\phi)$, $N(i,\phi) \wedge \neg B(i,\phi)$, $\neg N(i,\phi) \wedge B(i,\phi)$ and $\neg N(i,\phi) \wedge \neg B(i,\phi)$. In other respects, again unlike intention, an agent who needs two properties simultaneously, needs each of them: $\models N(i,\phi \wedge \psi) \Rightarrow N(i,\phi) \wedge N(i,\psi)$.

10 A FUNDAMENTAL PROPERTY OF THE LOGIC

The agents we have modelled, perfectly agree with themselves about *all* their own mental attitudes. This is

due to the subjective point of view and to the introspection abilities. So, an agent believes that she has (resp. has not) a given mental attitude, if and only if she has (resp. has not) this attitude, and conversely. Formally, the following general result can be proved:

Proposition 24: If ϕ is a formula where every proposition or predicate symbol appears in the scope of a modal operator formalizing a mental attitude of a given agent i, then:

(i) $\models \phi \Leftrightarrow B(i, \phi)$
(ii) $\models \phi \Leftrightarrow C(i, \phi)$
(iii) $\models B(i, \phi) \Leftrightarrow \neg B(i, \neg \phi)$

This proposition, which can be proved easily, exhibits the reason for which the choice made by an agent of any behavior about herself, is necessarily a relevant choice.

11 CONCLUSION

We have proposed a theory of intention which can be embedded into the model of an *autonomous* agent. Keeping C&L's methodological framework, we have highlighted some problems which arise in their theory and proposed solutions for them. We have adopted a subjective point of view (in order to give an agent autoepistemic abilities and, consequently, an own "existence") and provided original notions such as introspection-action, relevant choice or intention in a multiagent context. Moreover, in order to handle intention from a computational standpoint, we have introduced and formalized a new concept: need or potential intention, which bridges the gap between a declarative and an opeOrational semantics of intention.

References

J.F. Allen and R.C. Perrault (1980). Analyzing intention in utterances. *Artificial Intelligence* 15(3):143-178.

D. Appelt (1985). *Planning english sentences*. Cambridge University Press.

M.E. Bratman (1984). Two faces of intention. *The Philosophical Review* 93(3): 375-405.

M.E. Bratman (1987). *Intention, plans, and practical reason*. Harvard University Press.

P.R. Cohen and H.J. Levesque (1986). Persistence, intention and commitment. In M.P. Georgeff and A.L. Lansky (eds.), *Proceedings of the Timberline Workshop on "Reasoning about plans and actions*, 297-338. SRI International.

P.R. Cohen and H.J. Levesque (1990). Intention is choice with commitment. *Artificial Intelligence* 42(2-3): 213-262.

P.R. Cohen and H.J. Levesque (1991). Confirmations and joint action. In *Proceedings of twelfth IJCAI*: 951-957. Darling Harbour, Sidney, Australia.

P.R. Cohen and C.R. Perrault (1979). Elements of plan-based theory of speech acts. *Cognitive Science* 3(3): 177-212.

R. Fagin and H.J. Halpern (1987). Belief, awarness, and limited reasoning. *Artificial Intelligence* 34(1): 39-76.

J.Y. Halpern and Y. Moses (1985). A guide to the modal logics of knowledge and belief: a preliminary draft. In *Proceedings of ninth IJCAI*. Los Angeles, CA.

D. Harel (1984). Dynamic logic. In Gabbay, D., and Guentner, F. (eds.), *Handbook of philosophical logic, Volume II: Extensions of classical Logic*: 497-604. D. Reidel Publishing Company.

H.J. Levesque (1990). All I know: a study in autoepistemic logic. *Artificial Intelligence* 42(2-3): 263-309.

H.J. Levesque, P.R. Cohen, and J.H.T. Nunes (1990). On acting together. In *Proceedings of the eighth AAAI conference*, 94-99. Boston, MA.

R.C. Moore (1980). *Reasoning about knowledge and action*. Technical Note 191. Menlo Park, CA: SRI International, AI center.

C.R. Perrault and J.F. Allen (1980). A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics* 6(3): 167-182.

A.S. Rao and M.P. Georgeff (1991). Asymmetry thesis and side-effect problems in linear time and branching time intention logics. In *Proceedings of twelfth IJCAI*: 498-904. Darling Harbour, Sidney, Australia.

S.J. Rosenschein (1981). Plan synthesis: a logical perspective. In *Proceedings of the seventh IJCAI*: 331-337. Vancouver, BC.

M.D. Sadek (1990). Logical task modelling for Man-machine dialogue. In *Proceedings of the eighth AAAI conference*: 970-975. Boston, MA.

M.D. Sadek (1991a). *Attitudes mentales et interaction rationnelle: vers une théorie formelle de la communication*. Thesis dissertation. University of Rennes, France.

M.D. Sadek (1991b). Dialogue acts are rational plans. In *Proceedings of Venaco II workshop on "The structure of mutlimodal dialogue"*. Maratea, Italy.

J.R. Searle (1983). *Intentionality: an essay of philosophy of mind*. New York: Cambridge University Press.

VIII.

Diagnosis and Abduction

Knowledge Representation and Incorporation in a Hybrid System with Feedback

Yeona Jang
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
Yeona@medg.lcs.mit.edu

Abstract

This research investigates the advantages of introducing feedback between the processes of reasoning and knowledge acquisition in the diagnosis of multiple disorders in human patients. The introduction of such feedback results in an "hybrid" system that analyzes the results of previous diagnosis and incorporates their key features into an associative knowledge base which, in turn, assists future diagnosis.

In particular, this paper formally investigates a new mechanism, called a "diagnostic-unit" representation, for remembering the results of previous diagnosis. Unlike typical bipartite "If-Then" representations, the diagnostic-unit representation uses a general graph representation to capture more complex causal relationships between disorders and clusters of findings. The diagnostic-unit representation is explicitly tailored to facilitate the use of an educated decomposition-based abductive strategy in diagnosing disorders, by providing for a basis for remembering highly likely relationships between disorders and clusters of findings that are likely to be representative of high-level "macro-findings." This paper also investigates experience-guided strategies for incrementally deriving and updating diagnostic units and the various relationships among them. The results of previous diagnosis are remembered in a "decomposed-and-merged" form with respect to particular disorders.

1 Introduction

Patients suffering from multiple disorders are not rare in domains like congestive heart failure and Acquired Immune Deficiency Syndrome (AIDS). Unfortunately, less progress than expected has been made toward a system capable of competently and efficiently performing the diagnosis of multiple disorders. The slow advancement can be accounted for, largely, by the difficulty in balancing the issues of representation, knowledge acquisition, and problem solving. This paper focuses mainly on investigating a representational scheme for the diagnosis of multiple disorders, with an eye for achieving an effective balance between these issues.

[0]This research was supported in part by the National Heart, Lung, and Blood Institute under the grant number R01 HL33041 and in part by the National Institutes of Health under the grant number R01 LM04493 from the National Library of Medicine.

1.1 Motivations

One factor which hinders advancement in the development of a competent system for the diagnosis of multiple disorders is the need to provide accounts of underlying pathophysiologic mechanisms. While a single listing of the disorders primarily suspected of causing a given set of findings is computationally less burdensome to generate, oversimplified accounts of underlying causality provide few insights about how these disorders are producing the findings, and thus giving little guidance in providing therapy. It is particularly important to provide an underlying pathophysiologic mechanism, when medical therapy is unavailable either for a patient's complaints or for an elemental primary disease, but an effectively treatable "cause" may exist as an intermediate state on a causal pathway from a disease to findings. In addition, without a pathophysiologic mechanism that explains how findings are related to their primary suspected diseases, it is difficult to determine whether results of diagnosis "make sense" or are found by accident.

Advancement is also hampered by "knowledge inundation" which makes diagnosis from first principles computationally intensive. For instance, consider causal relationships between disorders and findings. Such relationships are often many-to-many, indirect, and uncertain. As knowledge is added, entities in a knowledge base are increasingly likely to interact in uncertain ways with each other. In an attempt to deal with the complexity of diagnosing multiple disorders, this research investigates an "hybrid" reasoning approach that seeks to combine the efficiency of association-based techniques [BuSh84, PGKS84, MPM84] and the robustness of model-based techniques [WKAS78, Pat81, Lon89]. The hybrid reasoning architecture investigated in this research makes use of a relatively robust causal-model-based component for diagnosis from first principles and an association-based component for diagnosing disorders that "have been seen before." In particular, this research attempts to abstract and organize complex pathophysiologic knowledge in forms

easily usable by the association-based component for the efficient diagnosis of multiple disorders.

What to abstract: The question that arises is, then, what knowledge to abstract. Unfortunately, what to abstract is not independent of the goal of problem solving, since it depends heavily on the use to which we intend to put the knowledge. Abstracting knowledge in an intentional vacuum is often too general to efficiently and effectively lead any particular problem solving to a solution.

This research attempts to abstract knowledge which allows search-intensive problem solving to be reduced to faster retrieval-based problem solving. To do so, this research observes that the context-sensitivity of findings can greatly affect overall diagnostic performance. In most medical domains, findings in isolation often have more than one cause, but findings as a whole can constrain each other's cause, consequently reducing the number of causes to consider. In addition, the significance of a finding depends on the other findings that occur together with it. As a consequence, changing some findings may even require findings that remain the same to be explained differently. For example, high cardiac output is often the most likely explanation for systolic ejection murmur. If findings that strongly suggest low cardiac output are present, however, aortic stenosis might be a better explanation for murmur than high cardiac output.

The main impetus of this research is the desire to capitalize on the context sensitivity of findings. This research presumes that findings can be grouped, or *clustered*, together in such a way that the findings in a set, taken together, immediately suggest not only their most likely disorder but also the pathophysiologic mechanism that best explains how this disorder is causing the findings. Such context-sensitive knowledge allows diagnosis of multiple disorders to be reduced to reasoning about how to decompose a set of findings, collected for diagnosis, into smaller subsets which can be solved immediately and relatively independently of each other.

Where to acquire: The next question is where to acquire rules to guide the finding decomposition process. A typical method is an interview with a domain expert. Cognitive and AI Researchers have found, however, that it is often hard to directly draw such experiential knowledge even from expert physicians [Pea88, TvKa77]. This research takes the view that problem-solving experience is the processes through which knowledge is structured into a ready-to-use coherently simplified whole. The intent is to use diagnostic experience, *i.e.*, results of previous diagnosis, to acquire operative associative knowledge. Relationships between disorders and findings that are critical to a current problem are highlighted in results of diagnosis, while insignificant ones are suppressed. The results of previous diagnosis, therefore, can act as knowledge that reduces search in future diagnosis. This observation motivates the conception of a system in which there is feedback between the reasoning and knowledge acquisition processes. The assessment of a hypothesis computed by problem solving becomes a key link in a feedback loop. In particular, such assessment gives notice of changes in knowledge, to which associative knowledge should be adjusted. The dynamic incorporation of revisions, *i.e.*, new experience, into an associative knowledge base allows the incorporation of changes prior to beginning a new problem-solving cycle. This adjustment, thus, feeds forward to future diagnosis.

How to remember: Because the choice of problem-solving algorithms and representational mechanisms often has a significant impact on overall problem-solving performance, the issue of how to remember results of previous diagnosis must be raised. Resolving this issue requires one to consider a variety of representational issues. For example, one needs to decide whether to treat each solved case as "atomic" or decomposable. This choice of a "grain size" can affect overall reasoning performance and domain understanding. The simplest approach is to store every solved case as an independent unit, as in case-based reasoning. While easy to implement, storing a case as an atom can limit the reusability of previous cases in future problem solving. Remembering every solved case independently has the following drawbacks. First, because even similar cases are stored separately, inefficient use of memory space can result. In addition, such "redundancy" can adversely affect overall problem-solving efficiency. For example, as solved cases are added to a system, we expect the system to be able to solve a problem without search, by directly retrieving a solution to a similar previous case. In deliberating about what case to use, the system must consider all the stored cases. If there are many cases, then simply matching them to find the best match(es) can be very time-consuming. On the other hand, if a diagnostic solution is decomposed into pieces that are too small, then the abstracted search space would become as big as the original knowledge base, and there might not be any gain in either reasoning performance or domain understanding.

This research approaches the issue of how to remember results of previous diagnosis, by asking the question "How can we make the results operational with respect to diagnostic problem solving?" By analyzing the results of previous diagnosis with respect to particular disorders, this research attempts to remember the results in a decomposed form which captures the context sensitivity of findings. In addition, solved cases are remembered not individually but in a "merged" form, where each component of a decomposed case can be merged with components of other solved cases to produce a coherent combined whole. Such merging only occurs, however, if the components

are instantiations of the same underlying pathophysiologic mechanism for a particular disorder.

The primary mechanism investigated by this research, for remembering results of diagnosis, is a "diagnostic-unit" representation. Each diagnostic unit is a general graph that represents an associative relationship between a disorder and a set of findings that, taken as a whole, strongly suggest the disorder. Unlike a typical bipartite representation, each associative relationship between a disorder and a set of findings is supported by a highly likely pathophysiologic mechanism that underlies the association. By clustering findings in such a way as to capture context sensitivity, the diagnostic-unit representation facilitates the effective use of an educated decomposition-based abductive strategy in diagnosis. The controlled clustering avoids sorting through a pile of patient findings, and facilitates the recognition of promising "macro-findings" that are solvable immediately and (relatively) independently of each other. Moreover, a focused commitment to capture the qualitative relationship of "the most likely" in diagnostic units, via relationships between them, allows proposing sensible causal-level diagnostic solutions, by combining partial solutions to these macro-findings.

This paper focuses primarily on the investigation of the diagnostic-unit representation and of issues on experiential-guided knowledge acquisition strategies for incrementally deriving and updating diagnostic units and the various relationships among them. The method investigated in this paper is discussed in the domain of the human cardiovascular system, even though they are domain-independent and should be applicable in other domains with similar characteristics.

Sections 2 and 3 discuss two types of knowledge representation, causal representation and diagnostic-unit representation, that this research uses in its hybrid problem solving. Sections 4 and 5 attempt to address issues that arise in incorporating new knowledge (experience) into an existing body of knowledge.

2 Causal Representation and Diagnosis

This section formally describes a causal representation of pathophysiologic knowledge, adopted from HF: HF is a probabilistic causal-model-based reasoner for differential diagnosis and management of patients with heart failure [LNCJ87, Lon89]. This section also discusses the complexity of clinical diagnosis based on causal representation. While causal representation is not new [Pea88, PR90], the formal investigation of a causal representation of medical knowledge facilitates the discussion of representational and acquisitional approaches explored in this research.

2.1 Causal Representation of Pathophysiologic Knowledge

The basic elements of the causal representation include elemental disorders, intermediate states, findings, and causal relationships between them. Elemental disorders and intermediate states are pathophysiologic states. Pathophysiologic states are states of living organisms, and their components, arising from bodily abnormality or failure to function properly. Elemental disorders are diagnostic states or primary causes. Diagnostic states are pathophysiologic states defined at the level needed to distinguish different manifestations, as defined in HF [LNCJ87, Lon89]. A primary cause is a pathophysiologic state that does not require further cause for it. Intermediate states are pathophysiologic states that are neither primary causes nor diagnostic states. Findings include patient history, subjective symptoms, and objective vital signs revealed either by observations or by various special laboratory tests, such as an X-ray examination.

Graphical notations are used to describe the basic elements of the causal representation. Let $\mathcal{D}$ denote a set of variables that represent elemental disorders, and $\mathcal{I}$ denote a set of variables that represent intermediate states. Let $\mathcal{F}$ stand for a set of variables that represent findings. A black rectangular node represents an elemental disorder variable in $\mathcal{D}$, an oval node an intermediate state variable in $\mathcal{I}$, and a rectangular node a finding variable in $\mathcal{F}$.

Causal mechanisms in a human body are often uncertain. Such uncertainty can be modeled in probabilistic terms, with a particular probability being used to represent the degree of belief in a causal dependency between states and/or findings.

Notation (Direct causal relation): *For any $a \in \mathcal{D} \cup \mathcal{I}$ and any $b \in \mathcal{D} \cup \mathcal{I} \cup \mathcal{F}$, such that a is a direct cause for b, let $a \xrightarrow{Pr(b|a)}_c b$ denote a direct causal relation from a to b, where given that a (and nothing else) occurred, a can cause b with probability of $Pr(b|a)$.*

A conditional probability associated with each causal link represents the strength of a cause in producing a particular effect. According to probability combining rules based on a noisy-or assumption, it can be computed, from the evidence collected, how likely it is that various disorders are present. (See [Lon89] for details.)

In this research, findings that can cause states or other findings are handled by creating corresponding pathophysiologic states. More specifically, for any finding f in $\mathcal{F}$ such that f can be a direct cause for a finding or state a, f is treated as if it is also a pathophysiologic state: A variable representing a corresponding pathophysiologic state i is created in either $\mathcal{D}$ or $\mathcal{I}$ accordingly, and a direct causal link

from i to f is established. In addition, instead of a direct causal relation from f to a, a direct causal relation from i to a' is established, where a' is a, if a is a state, or is a corresponding pathophysiologic state of a if a is a finding. For any direct cause b for f, a direct causal link from b' to i, rather than from b to f, is established, where b' is b if b is a state, or is a corresponding pathophysiologic state of b if b is a finding. Modeling findings that can cause states or other findings in this way guarantees that findings in $\mathcal{F}$ always appear as effects.

Let $\mathcal{L}$ be the set of all possible direct causal links, *i.e.*, $\mathcal{L} = \{a \xrightarrow{Pr(b|a)}_c b \mid a \in \mathcal{D} \cup \mathcal{I}, b \in \mathcal{D} \cup \mathcal{I} \cup \mathcal{F}$ and $\exists$ a direct causal relation from a to $b\}$.

Notation ($\mathcal{L}_{|D,I,F}$): *For any $D \subseteq \mathcal{D}, P \subseteq \mathcal{I}$, and $F \subseteq \mathcal{F}$, let $\mathcal{L}_{|D,I,F}$ denote the set of all possible probabilistic causal links between elements in D, P, and F. In other words, $\mathcal{L}_{|D,I,F} = \{a \xrightarrow{Pr(b|a)}_c b \mid a, b \in D \cup P \cup F$ and $\exists$ a direct causal relation from a to $b\}$.*

To facilitate further formal discussion, this research adapts a standard graph notation [CLR90] to define a causal graph. A *causal graph* G is defined as follows.

Definition 1 (Causal graph): *A causal graph G is a list (D, I, F, L), where $D \subseteq \mathcal{D}, I \subseteq \mathcal{I}, F \subseteq \mathcal{F}$, and $L \subseteq \mathcal{L}_{|D,I,F}$. D, I, F, and L are called the disorder set, intermediate state set, finding set, and causal link set, respectively, of G.*

A causal graph is a collection of direct causal dependencies between states and/or findings. Let $\mathcal{CG}$ be the universe set of all causal graphs: In other words, $\mathcal{CG} = \{(D, I, F, L) \mid D \subseteq \mathcal{D}, I \subseteq \mathcal{I}, F \subseteq \mathcal{F}$, and $L \subseteq \mathcal{L}_{|D,I,F})\}$.

Notation: *For any causal graph $G \in \mathcal{CG}$, let $d\langle G\rangle$, $f\langle G\rangle$, $i\langle G\rangle$, and $l\langle G\rangle$ denote the disorder set, finding set, intermediate state set, and causal link set, respectively, of G.*

Pathophysiologic knowledge that describes the malfunctioning of a human body can be conceptualized as a causal network of nodes and links, where each node represents a variable, [1] and each link represents a direct causal relationship between the states and/or findings represented by the nodes it connects. Throughout this proposal, the symbol $\mathcal{C}$ is used to denote a causal network that models pathophysiologic knowledge in a medical domain of concern. Causal network $\mathcal{C}$ is the causal graph $(\mathcal{D}, \mathcal{I}, \mathcal{F}, \mathcal{L})$.

[1] Except when noted otherwise, a node and a variable are used interchangeably.

2.2 Diagnosis of Multiple Disorders based on Causal Representation

This subsection discusses diagnosis from first principles, based on the causal representation described in Section 2.1. It also attempts to touch upon some related computational issues.

Additional notations are introduced to describe value assignments for variables. For any variable x in $\mathcal{C}$, let V_x denote a set of values that can be assigned to x. For any v in V_x, value assignment $x := v$ (for example, aortic stenosis := *present*) represents the instantiation of the value of x as v. An instantiated variable is a variable with a value assigned to it. A *finding* is an instantiated finding variable in $\mathcal{F}$, an *elemental disorder* an instantiated elemental disorder variable in $\mathcal{D}$, and an *intermediate state* an instantiated intermediate state in $\mathcal{I}$. This research assumes that for any variable x in $\mathcal{C}$, its values can be classified into two classes: normal and abnormal.

Definition 2 (Instantiated causal graph): *For any causal graph $G \in \mathcal{CG}$, G is said to be instantiated if for each node in G, the value of the node is instantiated.*

Similarly, an instantiated set can be defined as follows:

Definition 3 (Instantiated set): *For any variable set X, X is said to be instantiated if for any variable in X, the value of the variable is instantiated.*

Definition 4 (Equality of instantiated sets): *For any two instantiated variable sets X and Y, X and Y are said to be equal if*

1. *for any variable $x \in X$, $x \in Y$,*
2. *for any variable $y \in Y$, $y \in X$, and*
3. *for any variable $a \in X \cap Y$, the value of assignment of a in X is equal to that of a in Y.*

For expository purposes, a fictional causal network $\mathcal{C}$, shown in Figure 1, is used. Formally,

$\mathcal{C} = (\mathcal{D}, \mathcal{I}, \mathcal{F}, \mathcal{L})$, where
$\mathcal{D} = \{d_1, d_2, d_3\}$
$\mathcal{I} = \{i_1, i_2, i_3, i_4, i_5, i_6\}$
$\mathcal{F} = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7\}$
$\mathcal{L}$ = a set of direct causal links shown in Figure 1.

Assume that each node in $\mathcal{C}$ is a binary variable that can take on either *present* or *absent*. Further suppose that for any variable x in $\mathcal{C}$ *present* is an abnormal value of x, and *absent* is a normal value of x.

Let the symbol Ψ denote a diagnostic problem, *i.e.*, a set of findings collected for diagnosis: Ψ is an instantiated subset of $\mathcal{F}$. Ψ is also called a patient description. Diagnostic problem solving can be viewed

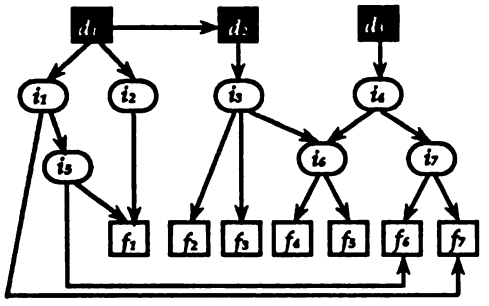


Figure 1: A fictional causal network C used for expository purposes

as the task of identifying an instantiation of C' that best explains the findings in Ψ, where C' is a causal network such that

1. C' is a subgraph of C that is obtained by removing from C nodes that represent findings not in Ψ, along with any direct causal links to these nodes, and
2. $f\langle C'\rangle$ is an instantiated set equal to Ψ.

This research calls C' "a tailored causal network of C to diagnostic problem Ψ " (or simply "a tailored causal network of C "). Note that the modeling, as pathophysiologic states, of findings that can cause other states or findings guarantees that all findings in $\mathcal{F}$ always appear as leaves in C. As a result, the finding removal process to produce a tailored causal network of C can be performed without considering issues of how to remove nodes with children. For example, suppose that a diagnostic problem Ψ consists of findings $f_3 :=$ *present*, $f_4 :=$ *present*, $f_6 :=$ *present*, and $f_7 :=$ *present*. Figure 2 is a tailored causal network of C to Ψ.

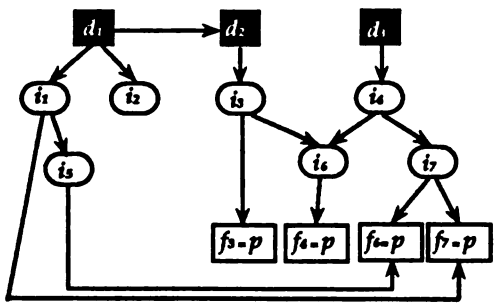


Figure 2: A causal network tailored to diagnostic problem Ψ that consists of $f_3 :=$ *present*, $f_4 :=$ *present*, $f_6 :=$ *present*, and $f_7 :=$ *present*

The following notation is introduced to facilitate the discussion of a diagnostic solution.

Notation ($di\langle G\rangle, dif\langle G\rangle$): *For any causal graph $G \in \mathcal{CG}$, $di\langle G\rangle$ is the union of $d\langle G\rangle$ and $i\langle G\rangle$, and $dif\langle G\rangle$ is the union of $d\langle G\rangle$, $i\langle G\rangle$, and $f\langle G\rangle$: In other words, $di\langle G\rangle = d\langle G\rangle \cup i\langle G\rangle$, and $dif\langle G\rangle = d\langle G\rangle \cup i\langle G\rangle \cup f\langle G\rangle$.*

The symbol $\cup$ used in the above notation represents a union of variable sets which denotes the *conjunction*, not logical disjunction, of events asserted by instantiating the variable set union. For example, consider a variable set A which consists of a variable a which is instantiated as v_a, and a variable set B which consists of a variable b with v_b assigned to it. $Pr(A \cup B)$ stands for the probability that a is instantiated as v_a, and b as v_b – in other words, $Pr(a := v_a, b := v_b)$.

Definition 5 (Causal explanation): *For some instantiated subset F of $\mathcal{F}$, a causal explanation for the findings in F is an instantiation of a tailored causal network of C to F.*

Due to uncertainty in the domain knowledge itself, it is often difficult to find a solution to a diagnostic problem with absolute certainty. For such problems, probabilities that summarize which disorders are more likely than others can be used to support diagnostic judgments. Observe that any instantiation of a tailored causal network of C to a diagnostic problem Ψ can be an explanation for the findings in Ψ. Each of these causal explanations tells us that the pathophysiologic states in the causal explanation can cause the findings accordingly, but not necessarily do so. It is critical to find the most likely causal explanations, particularly when high risk and costs are associated with treatments. A *solution* to Ψ is defined as a causal explanation S^*_Ψ, for Ψ, that satisfies the qualitative relationship that for any causal explanation G for Ψ, the findings in Ψ are more, or equally, likely to be caused by the disorders in $d\langle S^*_\Psi\rangle$ via the pathophysiologic mechanism identified by S^*_Ψ than by those in $d\langle G\rangle$ via the pathophysiologic mechanism identified by G. This relationship can be articulated as follows.

Definition 6.1 (Ideal diagnostic solution): *For any diagnostic problem Ψ, the ideal diagnostic solution to Ψ is a causal explanation S^*_Ψ for Ψ, such that for any causal explanation G for Ψ, $Pr(di\langle S^*_\Psi\rangle \mid \Psi) \geq Pr(di\langle G\rangle \mid \Psi)$.*

Despite of the use of tailoring, C and its instantiations are often too large to convey useful information to a user. A user of a diagnostic system is usually more interested in abnormal states that produce a given set of findings. To make a diagnostic solution more informative and insightful, this research attempts to trim the ideal diagnostic solution in Definition 6.1 by pruning away nodes with normal values. The following defines a reduced ideal diagnostic solution.

Definition 6.2 (Reduced ideal diagnostic solution): *For any diagnostic problem Ψ, the reduced ideal diagnostic solution to Ψ is a maximal subgraph G of the ideal diagnostic solution S^*_Ψ, such that*

1. *$dif\langle G\rangle$ is a subset of $dif\langle S^*_\Psi\rangle$ such that every node in G is instantiated to an abnormal value,*
2. *every node in $dif\langle S^*_\Psi\rangle - dif\langle G\rangle$ is instantiated to a normal value, and*
3. *$l\langle G\rangle = \{a \rightarrow b \mid a \in di\langle G\rangle, b \in dif\langle G\rangle$, and $\exists$ a direct causal relation from a to b in $S^*_\Psi\}$.*

Suppose, for example, that the instantiated, tailored causal network in Figure 3 is the most likely explanation for $f_3 :=$ *present*, $f_4 :=$ *present*, $f_6 :=$ *present*, and $f_7 :=$ *present*.

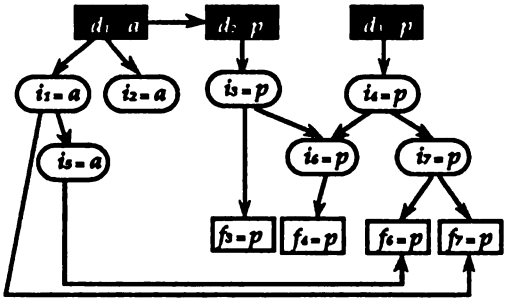


Figure 3: The most likely instantiation of the tailored causal network of C in Figure 2

Then, the reduced ideal diagnostic solution to our diagnostic problem is the "trimmed" best instantiation shown in Figure 4. For convenience, reduced ideal diagnostic solutions are generally referred to as "ideal diagnostic solutions," hereinafter.

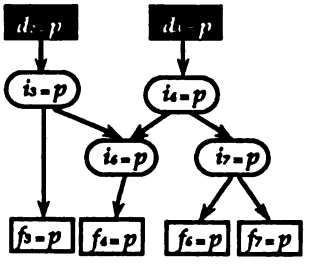


Figure 4: The ideal diagnostic solution obtained by trimming the best instantiation in Figure 3

Note that a highly likely pathophysiologic mechanism is returned as a part of a diagnostic solution. A pathophysiologic mechanism in a diagnostic solution can provide additional assurance by explaining how the primary disorders and findings are related to each other. In addition, the underlying pathophysiologic mechanism in a diagnostic solution makes it easier to reason about the solution and determine if the solution found by a program "makes sense." Finally, the underlying pathophysiologic mechanism in a diagnostic solution makes it easier to detect possible sources of flaws in a diagnostic solution. This will, in turn, help confirm or refine the correctness of the knowledge base used by a program.

Given a set of findings collected for diagnosis, a probabilistic causal-model-based reasoner, such as HF [Lon89], evaluates the state of the causal network by tracing backward through causal paths from each finding. Reasoning in the causal network amounts to the propagation of impacts of findings throughout the network.

The prevalence of multiple uncertain causations, however, makes the computation non-trivial. Several causal paths, none of which are certain, often come together at a node, and the number of potential explanatory paths to consider grows exponentially in the number of nodes that come together at a node. The exhaustive enumeration-and-evaluation approach is, thus, likely to be unwieldy.

Fortunately, the search can be reduced by using heuristics. For example, HF uses heuristics such as precomputing and pruning to relieve computational complexity [LNCJ87]. Despite the use of such heuristics, however, HF must calculate an estimated probability for every path, and combine exponentially many relevant pathways. This research attempts to lessen the computational complexity of causal-model-based diagnosis, by incrementally abstracting, from relatively unstructured causal knowledge, patterns that cluster findings according to their most likely disorders and underlying pathophysiologic mechanisms. Such clusters can then be remembered for use in future diagnosis.

3 Diagnostic-Unit Representation

This section discusses the diagnostic-unit representation of knowledge abstracted from causal knowledge. The primary constructs of the diagnostic-unit representation are diagnostic units and links between them.

3.1 Diagnostic Units

Diagnostic units attempt to cluster findings according to their most likely disorders and pathophysiologic mechanisms. Additional definitions and notations are given below to facilitate the discussion of diagnostic units.

Definition 7 (Source set of a causal graph): *For any causal graph $G \in \mathcal{CG}$, the source set of G, denoted by $s\langle G\rangle$, is the set of nodes on G which do not have incoming links in G.*

Definition 8 (Ultimate diagnostic unit $du^*(d,F)$): *For some instantiated subset F of $\mathcal{F}$ and some $d \in \mathcal{D}$, an ultimate diagnostic unit $du^*(d,F)$ of d is an instantiated subgraph of $\mathcal{C}$ such that*

1. $s\langle du^*(d,F)\rangle = \{d\}$,
2. $f\langle du^*(d,F)\rangle = F$, *and*
3. *for any instantiated subgraph G of $\mathcal{C}$ such that $f\langle G\rangle = F$,*
   $Pr(di\langle du^*(d,F)\rangle \mid F) \geq Pr(di\langle G\rangle \mid F)$.

In other words, an ultimate diagnostic unit $du^*(d,F)$ is a causal explanation, for F, that is rooted at a single source d, and satisfies the qualitative relationship such that for any causal explanation G for F, the findings in F are more, or equally, likely to be caused by d via the pathophysiologic mechanism identified by $du^*(d,F)$ than by the disorders in $d\langle G\rangle$ via the pathophysiologic mechanism identified by G.

In principle, for any combination of an elemental disorder and a subset of $\mathcal{F}$, the corresponding ultimate diagnostic unit can be computed. The issue is one of computational resources. The number of subsets of $\mathcal{F}$ to consider grows exponentially in the number of variables in $\mathcal{F}$. To make matters worse, for each subset F of $\mathcal{F}$, there are exponentially many supersets of F to consider. This research attempts to deal with such combinatorial explosions by acquiring diagnostic units that appear empirically useful. In particular, it views diagnostic problem solving as a process by which ultimate diagnostic units and relationships between are gradually recovered. Section 5 discusses issues that arise in incrementally recovering ultimate diagnostic units on an empirical basis.

Different sets of findings can suggest the same elemental disorder via different underlying pathophysiologic mechanisms. For example, myocardial infarction frequently produces systolic ventricular emptying dysfunction with relatively high likelihood. Myocardial infarction can also be a highly likely cause of diastolic ventricular filling dysfunction – though with less frequency. Each of the dysfunctions has a different underlying pathophysiologic mechanism, and thus, corresponds to a different diagnostic unit. In light of the possibility that an elemental disorder can have multiple diagnostic units, the set of ultimate diagnostic units for an elemental disorder is defined as follows:

Definition 9 (Ultimate diagnostic-unit set $DU^*(d)$): *For some $d \in \mathcal{D}$, the ultimate diagnostic-unit set of d, denoted by $DU^*(d)$, is the set of ultimate diagnostic units of d.*

3.2 Links between Diagnostic Units

The definition of an ultimate diagnostic unit does not mention how an ultimate diagnostic unit is related to other ultimate diagnostic units. This subsection defines two types of links that represent relationships between ultimate diagnostic units. One type of link represents a causal dependency between diagnostic units.

Definition 10 (Causal relation between diagnostic units): *For any two diagnostic units $du^*(d_1)$ and $du^*(d_2)$, $du^*(d_1)$ is said to be causally related to $du^*(d_2)$, denoted by $du^*(d_1) \rightarrow_c du^*(d_2)$, if there exist an ideal diagnostic solution S^*_Ψ such that*

1. *$du^*(d_1)$ and $du^*(d_2)$ are subgraphs of S^*_Ψ, and*
2. *there exists a causal path W in S^*_Ψ from d_1 to d_2 such that d_1 and d_2 are the only elemental disorders in W.*

The other type of link represents a dependency between diagnostic units which are not causally related but still share a common node.

Definition 11 (Non-causal relation between diagnostic units:) *For any two diagnostic units $du^*(d_1)$ and $du^*(d_2)$, $du^*(d_1)$ is said to be non-causally related to $du^*(d_2)$, denoted by $du^*(d_1) \rightarrow_\ell du^*(d_2)$, if there exist an ideal diagnostic solution S^*_Ψ such that*

1. *both $du^*(d_1)$ and $du^*(d_2)$ are subgraphs of S^*_Ψ,*
2. *there exists no causal path in S^*_Ψ either from d_1 to d_2 or from d_2 to d_1, and*
3. *there exists a node n such that there exist in S^*_Ψ a causal path W_1 from d_1 to n and a causal path W_2 from d_2 to n such that d_1 and d_2 are the only elemental disorders in W_1 and W_2, respectively.*

When two diagnostic units are related to each other both causally and non-causally, a causal relation link is established between them.

Knowledge represented in the diagnostic-unit representation can be conceptualized as a graph where each node represents a diagnostic unit set, and each link represents a relationship between diagnostic units in different diagnostic-unit sets. Such a graph is called an *diagnostically-operative causal graph* (or simply DOC), and is defined as follows:

Definition 12 (Diagnostically-operative causal graph): *A diagnostically-operative causal graph G is a pair (CU, L), where CU is a set of diagnostic unit sets and L is a set of links between diagnostic units in the diagnostic-unit sets in CU.*

Each ultimate diagnostic unit represents the most likely causal relation between a cluster of findings and a disorder. Findings in a diagnostic unit, taken together, form a distinct clinical indicator of a disorder and a corresponding particular pathophysiologic mechanism. In addition, a diagnostic unit captures a diagnostic context where the occurrence of the findings in the diagnostic unit suggests, (relatively) independently of findings not in the diagnostic unit, the disorder and the corresponding pathophysiologic mechanism identified by the unit. "The most likeliness," embedded in diagnostic units, allows causal dependencies critical to a diagnostic situation to be quickly identified and readily accessed. If diagnostic units are available, then future diagnosis can potentially generate causal explanations for findings abductively, without having to duplicate comparatively expensive reasoning.

4 Experience and Acquisition of Diagnostic Units

Once ultimate diagnostic units are available, a solution to a diagnostic problem can be proposed by selecting and combining the diagnostic units relevant to the problem. In addition, diagnostic units can enhance understanding of the vital relationships between disorders and findings. The issue that must be dealt with is, then, where and how to acquire ultimate diagnostic units. As defined in Section 2.2, the results of diagnosis highlight causal dependencies critical to a given problem, while pruning away insignificant ones, from the detailed causal knowledge in $\mathcal{C}$. Consider a diagnostic solution S_Ψ computed by a system such as HF, for some set Ψ of findings collected for diagnosis. Diagnostic solution S_Ψ is a causal explanation, for Ψ, such that for any (or most, if heuristics are used) causal explanation G for Ψ,

$$\widehat{Pr}(di\langle S_\Psi\rangle \mid \Psi) \geq \widehat{Pr}(di\langle G\rangle \mid \Psi),$$

where $\widehat{Pr}(a \mid b)$ denotes a probability estimated by the system that generated S_Ψ. In other words, diagnostic solution S_Ψ is an approximation of the ideal diagnostic solution $S^\star_\Psi$ to Ψ. This feature suggests the possibility of using computed diagnostic solutions as a source of incrementally recovering ultimate diagnostic units and relationships between them.

This research attempts to remember solved cases in a decomposed form by analyzing them with respect to elemental disorders. Each diagnostic solution $S^\star_\Psi$ is decomposed into smaller instantiated causal graphs each of which is obtained by collecting all nodes and links in $S^\star_\Psi$ that are reachable from each elemental disorder d in $d\langle S^\star_\Psi\rangle$. For example, consider again the diagnostic solution $S^\star_\Psi$ in Figure 4. $S^\star_\Psi$ can be decomposed into the components shown in Figure 5. Each component is a subgraph of $S^\star_\Psi$ rooted at a particular elemental disorder in $d\langle S^\star_\Psi\rangle$. Since $d\langle S^\star_\Psi\rangle$ consists of two elemental disorders, $S^\star_\Psi$ is decomposed into exactly two components.

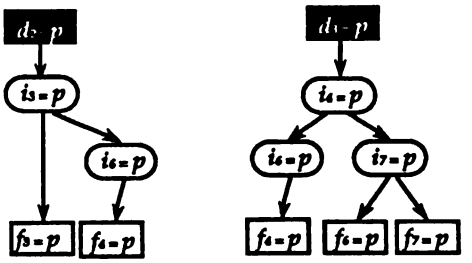


Figure 5: Components of $S^\star_\Psi$ in Figure 3 resulted from the decomposition

This research also investigates an approach to merge the components of a solved case with components of other solved cases to produce a coherent "whole" that is remembered for use in future diagnosis. Components of solved cases can be merged if they are instantiations of the same underlying pathophysiologic mechanism of a particular elemental disorder. The coherent combined whole is, then, remembered. By remembering similar cases in a merged form, resources such as memory space and processing time can be used more efficiently. The diagnostic-unit representation is the mechanism that provides a basis for remembering solved cases in a decomposed and merged form. Section 5 discusses in greater detail issues that arise in merging.

5 Knowledge Incorporation

This section investigates a method for using diagnostic experience as a source of recovering ultimate diagnostic units and the relationships between. This research assumes that knowledge acquired from experience is stored in an associative knowledge base, K_{AS}, that is initially empty. In other words, no diagnostic units are known *a priori*. As diagnostic experience grows, however, empirically useful ultimate diagnostic units and relationships between them are gradually recovered. Partially recovered diagnostic units and relationships between them are stored in K_{AS} to assist future diagnosis.

Whenever a diagnostic problem is solved, a "knowledge-incorporating" process assimilates diagnostic experience into K_{AS} by transforming the diagnostic solution to a DOC, and then by incorporating it into K_{AS}. The rest of this section discusses the knowledge-incorporating process, and briefly addresses issues that affect the process.

5.1 Diagnostically-Operative Causal Graph (DOC) Transformation Process

The following additional definition is introduced to facilitate further discussion.

Definition 13 (Restricted finding set $f\langle d|G\rangle$): *For any causal graph $G \in \mathcal{CG}$ and any $d \in d\langle G\rangle$, the finding set of d restricted to G, denoted by $f\langle d|G\rangle$, is the set of findings in G that are reachable from d. Thus, $f\langle d|G\rangle \subseteq f\langle G\rangle$.*

For each diagnostic solution S_Ψ generated by $\mathcal{S}$, the corresponding DOC of S_Ψ is constructed as follows.

Step I: For each d in $d\langle S_\Psi\rangle$, compile an approximate diagnostic unit $du(d, f\langle d \mid S_\Psi\rangle)$ by collecting all nodes and links in S_Ψ reachable from d.

Step II: For any two approximate diagnostic units compiled by Step I, establish either a causal or a non-causal link between them (See Definitions 10 and 11).

For example, consider the diagnostic solution S^*_Ψ in Figure 4. Step I produces two approximate diagnostic units which are rooted at d_2 and d_3, respectively, as shown in Figure 5. Step II completes the construction of a DOC for S^*_Ψ, by establishing a non-causal link between these two approximate diagnostic units.

5.2 Joining-Up Process

The process of assimilating diagnostic solutions into K_{AS} is called the *joining-up* process. The following types of experiential knowledge can be incorporated into K_{AS} by the joining-up process.

- New compiled diagnostic units and relationships between them.
- Statistical information such as the frequency with which findings in a diagnostic unit occur in support of the corresponding elemental disorder.

To facilitate further discussion, the following additional definition is introduced.

Definition 14 (Union of causal graphs): *For any set of causal graphs $G_1, G_2, \ldots,$ and G_n, where $G_i = (D_i, P_i, F_i, L_i)$, the union of the G_i's is defined as follows:*

$$\bigcup_{i=1}^{n} G_i = (\bigcup_{i=1}^{n} D_i, \bigcup_{i=1}^{n} P_i, \bigcup_{i=1}^{n} F_i, \bigcup_{i=1}^{n} L_i).$$

Let DOC_Ψ denote the diagnostically-operative causal graph produced by the preceding DOC transformation process. Let $\widehat{du}(d)$ and $\widehat{DU}(d)$ stand for a diagnostic unit and a diagnostic unit set, respectively, stored in K_{AS}. For each diagnostic unit $du(d)$ in DOC_Ψ, the following algorithm describes, at a high level, how $du(d)$ is incorporated into K_{AS}:

- If $\widehat{DU}(d)$ exists in K_{AS}, then
  - if there exists $\widehat{du}(d)$ in $\widehat{DU}(d)$ that $du(d)$ can be incorporated into, then modify the structure of $\widehat{du}(d)$ by unioning $\widehat{du}(d)$ and $du(d)$, and updating statistical information. The union of causal graphs is defined in Definition 14.
  - Otherwise, update $\widehat{DU}(d)$ by adding $du(d)$ to the set.
- If no $\widehat{DU}(d)$ exists in K_{AS}, then create $\widehat{DU}(d)$ with $du(d)$ as its element.

The incorporation of new diagnostic units raises the issue of determining whether a new diagnostic unit should be used to update an existing diagnostic unit in K_{AS} or be considered to be a new element of the corresponding diagnostic unit set. In other words, the question that needs to be answered is whether or not $du(d)$ and $\widehat{du}(d)$ should be considered to be both a partial recovery of the same ultimate diagnostic unit. The resolution of this issue is affected by the issue of what would be appropriate sizes for individual diagnostic unit sets (and, consequently, individual diagnostic units). Since the sizes of diagnostic unit sets (and of diagnostic units) can affect overall computational efficiency, it may be possible to trade off among reasoning efficiency, accuracy of solutions, and understanding of the vital relationships between disorders and findings.

Dependencies between diagnostic units are also added into K_{AS}. In addition, the joining-up process performs bookkeeping for use in later decomposition-based abductive diagnosis. In general, it is difficult to know in advance how much weight should be attached to each finding in a diagnostic unit. Because this weight represents the "strength" with which a finding supports the presence of the corresponding diagnostic unit, it is important that the weight be empirically well justified. Rather than pre-setting a certain fixed subjective estimate for each weight, this research investigates the use of frequency of occurrences as an estimate of a correct weight. As diagnostic experience grows, the joining-up process updates the frequency with which an elemental disorder occurs and the frequency with which a finding occurs in support of the elemental disorder. Weights of findings which appear to be good predictors of a diagnostic unit, then, will increase over time. Such statistical information provides useful guidance for recognizing the diagnostic units that are relevant to a problem.

After a diagnostic solution is incorporated, the chunks of associative knowledge remain in the associative memory, ready for use in future diagnosis. Diagnosis can, thus, be made by selecting and combining diagnostic units relevant to a current problem.

A preliminary experiment was conducted to assess the feasibility of diagnostic-unit representation.

In this experiment, 300 cases were solved by HF. For each HF-generated solution, the corresponding diagnostically-operative causal graph was constructed. A newly compiled approximate diagnostic unit on the diagnostically-operative causal graph was used to update an existing approximate diagnostic unit, if the structures of the two units match by more than 80 percent. This simple experiment shows that the average size of a diagnostic unit set is 3.1 diagnostic units, and that each diagnostic unit can have causal and/or non-causal relation links to, on average, seven other diagnostic units.

5.3 A Property of Knowledge in K_{AS}

It was claimed previously that diagnostic units acquired through diagnostic experience can be used as approximations of ultimate diagnostic units. The following theorem describes a property of diagnostic units stored in K_{AS}.

Theorem 1: *Diagnostic units in K_{AS} are approximations of ultimate diagnostic units.*

Proof: For all $\widehat{du}(d)$ in K_{AS}, $\widehat{du}(d)$ is an instantiated subgraph of $\mathcal{C}$ with a single root of d in $\mathcal{D}$. For expository simplicity, let $f(\widehat{du}(d)) = F$. We would like to prove that for most instantiated subgraphs G of $\mathcal{C}$ such that $f(G) = F$, $Pr(di(\widehat{du}(d)F) \mid F) \geq Pr(di(G) \mid F)$. The definition of probability based on the notion of relative frequency is employed to prove this. According to past diagnostic experience of $\mathcal{S}$, none (few, if any) of such G has (have) occurred so far. Therefore, for most instantiated subgraphs G of $\mathcal{C}$ such that $f(G) = F$, $Pr(di(\widehat{du}(d)) \mid F) \geq Pr(di(G) \mid F)$. Because of the lack of diagnostic experience, it is possible that other diagnostic units whose finding sets are equal to F can exist in K_{AS}. ■

Diagnostic units in K_{AS} are expected to be volatile at first but to be relatively stable in the longer term. The preliminary experiment appears to empirically support that diagnostic units reach their (relatively) ultimate values, as the number of occurrences becomes large.

5.4 Explanation-Based Learning

It is possible to use explanation-based learning to learn disease concepts. Explanation-based learning is a technique for learning a concept without the use of many training examples [GeMo86, MCKKEG89]. It consists of two steps: generation of an explanation for a given example, followed by generalization of the explanation. A domain theory is used to generalize the explanation, and it is the domain theory that makes it possible to learn a concept from only a few training examples. The direct application of explanation-based learning to disease concept learning, however, would be problematic without appropriate extensions to handle the following difficulties.

First, explanation-based learning is generally only effective in domains where pruning is simple. If a domain theory consists of causations where each effect has a single certain corresponding cause, the generation of an explanation is a relatively simple task: The cause of an effect can be identified with certainty. If there are many "levels" of uncertain multiple causations, however, exponentially many potential explanations need to be considered to generate a good explanation. The issue of how to deal with the combinatorial explosion that arises in generating an explanation, thus, emerges as an overriding concern.

Second, explanation-based learning generally assumes that a complete domain theory is available. This assumption is motivated by the desire to generalize explanations into provably correct ones. It is hard, if even possible, to come by a complete domain theory in ill-understood domains like medicine. In such domains, learning empirically justifiable experience is likely to be more sensible.

Third, knowledge learned using explanation-based learning can generally only be applied when exact matches occur. Within the context of medical diagnosis, this restriction implies that a learned disease concept cannot be used in diagnosis, unless a diagnostic problem matches the concept exactly. In most medical domains, it is not uncommon that patients can manifest different sets of findings, even when they are suffering from the same disorder. This characteristic of medical diagnosis suggests that partial matching may have greater utility.

Finally, much of explanation-based learning focuses primarily on learning concepts one at a time, independently of each other. In most medical domains where multiple disorders are not infrequent, however, a disease can have different descriptions, depending on what other diseases occur with it. Dependencies among concepts have to be handled appropriately if useful disease concepts are to be learned.

6 Related Work: Integration of Problem solving and Learning

What to learn is not independent of the problem solving goal. In the light of the dependency between problem solving and learning, systems such as SOAR coupled learning to problem solving. SOAR is a rule-based general-purpose problem-solving system which is integrated with explanation-based learning [LRN86]. An explanation-based learning component of SOAR analyzes explanations, and chunks macro rules that summarize the explanations [GeMo86, KeKC86].

Another general purpose problem-solving system which incorporates several learning mechanisms is PRODIGY. A good deal of learning in PRODIGY is di-

rected at automatically acquiring control rules from experience, to enhance the efficiency of a search process [MCKKEG89]. PRODIGY differs from SOAR in that it attempts to learn from its failures as well as successes. If pursuing an unsuccessful path, the PRODIGY seeks to come up with an explanation for the failure. This explanation is, then, used to construct control rules that will help PRODIGY avoid unprofitable search paths in later problem solving.

CASEY [Kot89] is a diagnostic system that acquires diagnostic knowledge from experience to improve performance in later diagnosis. It is a case-based, more specifically comparison-based, reasoning system grounded on the causal-model-based HF. It remembers cases that have been successfully solved, for use in future problem solving. It solves a new diagnostic problem by directly inspecting old cases to find best matches against the new one.

7 Concluding Remarks

This paper has described a framework for structuring diagnostically critical knowledge as graphs. Each graph is such that the causal explanation identified by the graph can immediately be inferred to be the best causal explanation for the findings on the graph. Possible strategies for incrementally acquiring such graphs by analyzing results of diagnosis with respect to disorders have been described.

Automated diagnosis of multiple disorders is an intriguing domain from both medical and artificial intelligence perspectives. From a medical perspective, patterns of findings can be useful indicators of disorders and corresponding underlying causal mechanisms. Expert physicians seem to examine findings for such indications. A computer program that can identify such patterns for diagnosis, thus, has great utility to physicians. This research will help discover and/or better understand the relationships, between findings and causes, critical to diagnosis.

From an artificial intelligence perspective, diagnosis of multiple disorders is a challenging area for addressing complexity and issues, in particular, interdependent issues of knowledge representation, knowledge acquisition, and reasoning efficiency. Research is being conducted at the MIT Clinical Medical Decision Making Group to design and implement knowledge assimilation strategies and decomposition-based abductive reasoning algorithms that more completely address such issues and trade-offs.

Acknowledgments

I would like to thank Peter Szolovits for his support and stimulating comments, William Long for his help in understanding HF codes, and Alexander Ishii for his heroic draft-reading efforts and numerous insightful suggestions.

References

[BuSh84] B.G. Buchanan, and E.H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, MA, 1984.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[GeMo86] G. Dejong and R. Mooney. "Explanation-based learning: An alternative view." *Machine Learning*, 1(2), 1986.

[KeKC86] R.M. Keller and S.T. Kedar-Cabelli. "Explanation-based generalization: A unifying view." *Machine Learning*, 1(1), 1986.

[Kot89] P.A. Koton. "Using experience in learning and problem solving." TR 441, Massachusetts Institute of Technology, Laboratory for Computer Science, Cambridge, MA, 1989.

[LRN86] J.E. Laird, P.S. Rosenbloom, and A. Newell. "Chunking in SOAR: The anatomy of a general learning mechanism." *Machine Learning*, 1, pages 11-46, 1986.

[LNCJ87] W.J. Long, S. Naimi, M.G. Criscitiello, and R. Jayes. "The development and use of a causal model for reasoning about heart failure." In *Symposium on Computer Applications in Medical Care*, pages 30–36, IEEE, 1987.

[Lon89] W.J. Long. "Medical diagnosis using a probabilistic causal network." *Applied Artificial Intelligence*, 3:367-383, 1989.

[MPM84] R.A. Miller, H.E. Pople, Jr., and J.D. Myers. "INTERNIST-1: An Experimental Computer-Based Diagnostic Consultant for General Internal Medicine." In W.J. Clancey and E.H. Shortliffe, editors, *Readings in Medical Artificial Intelligence*, pages 190–209, Addison-Wesley, Reading, MA, 1984.

[MCKKEG89] S. Minton, J.G. Carbonell, C.A. Knoblock, D.R. Kuokka, O. Etzioni,

and Y. Gil. "Explanation-based Learning: Optimizing Problem Solving Performance through Experience." *Paradigms for Maching Learning*, 1989.

[Pat81] R.S. Patil. "Causal representation of patient illness for electrolyte and acid-base diagnosis." MIT TR-267, MIT Lab. for Comp. Sci., 1981.

[PGKS84] S.G. Pauker, G.A. Gorry, J.P. Kassirer, and W.B. Schwartz. "Towards the Simulation of Clinical Cognition: Taking a Present Illness by Computer." In W.J. Clancey and E.H. Shortliffe, editors, *Readings in Medical Artificial Intelligence*, pages 131–159, Addison-Wesley, Reading, MA, 1984.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, San Mateo: Morgan Kaufmann, 1988.

[PR90] Y. Peng and J. Reggia. *Abductive Inference Methods for Diagnostic Problem Solving*, Springer Verlag, 1990.

[TvKa77] A. Tveretzky and D. Kahnemann. "Causal Schemata. in judgments under uncertainty." In *Progress in Social Psychology*, ed. M. Fishbein. Hillsdale, N.J., Lawrence Erlbaum, 1977.

[WKAS78] S.M. Weiss, C.A. Kulikowski, S. Amarel, and A. Safir. "A Model-Based Method for Computer-Aided Medical Decision Making." In *Artificial Intelligence* Vol. 11, pages 145-172, 1978.

Choosing Observations and Actions in Model-Based Diagnosis/Repair Systems

Gerhard Friedrich and Wolfgang Nejdl
Technische Universität Wien
Christian Doppler Labor for Expert Systems
Paniglgasse 16, A-1040 Vienna, Austria
e-mail: {friedrich,nejdl}@vexpert.dbai.tuwien.ac.at

Abstract

Common model-based diagnosis systems assume a separation of diagnosis and repair. More recent work showed the potential of exploiting repair knowledge already during the diagnosis phase. We show how to integrate this knowledge into diagnosis and repair plans minimizing overall diagnosis and repair costs. Our cost model includes time-dependent breakdown costs as well as observation and repair costs. Based on this cost evaluation we develop a set of polynomial-time greedy algorithms which transform conventional plans (without sensing procedures) into diagnosis and repair plans which include both actions for repair and observations for diagnosis. These methods are applied to a real world example (power transmission networks) showing the benefits compared to the conventional approach.

1 Introduction

We have previously stressed the need for integrating repair actions into the diagnosis process of model-based reasoning systems. During such a diagnosis and repair process an intelligent agent has to produce and evaluate repair schedules including both observations and actions. In this paper we will discuss how to choose the order of actions and observations based on the idea of minimizing breakdown and diagnosis/repair costs.

Recent work in planning [Chrisman and Simmons, 1991], where the agent has no exact state description, uses a fixed sensing procedure to avoid the enormous increase in complexity introduced by integrating sensing procedures into conventional plans. On the other hand model-based diagnosis has developed efficient techniques which choose the best observation dynamically depending on the current diagnoses to gain more information about the state of the world [de Kleer and Williams, 1987, de Kleer, 1991] but do not consider any repair actions during the diagnosis process.

We develop a set of greedy algorithms integrating both sensing and repair actions into a diagnosis repair plan, which improves the costs of the diagnosis/repair process compared to conventional model-based diagnosis, yet neither leads to an increase in complexity in the planning phase nor in the measurement/action selection phase (i.e. planning complexity is the same as without sensing and selection complexity is the same as without actions).

Using a cost estimation function based on such diagnosis/repair plans including time-dependent breakdown costs as well as diagnosis and repair costs, we are able to decrease overall costs. We reduce breakdown costs by repairing before diagnosis is completed and reduce diagnosis costs by distinguishing between diagnoses only when necessary. By clustering diagnoses depending on their repair plans, measurement selection is guided not only by minimizing measurement costs, but by minimizing breakdown/measurement/repair costs.

In Section 2 we describe a diagnosis/repair plan interleaving observation and repair actions to minimize overall costs in a defect power transmission network. Section 3 discusses the basic concepts of our approach (as defined in [Friedrich *et al.*, 1990, Friedrich *et al.*, 1991, Friedrich *et al.*, 1992, Nejdl, 1991]) and the architecture of our diagnosis/repair system. Section 4 formalizes Diagnosis/Repair Plans and a cost estimation function based on time-dependent breakdown costs and diagnosis/repair costs. Section 5 discusses the initial example again to illustrate our method.

2 Example

2.1 Domain Description

As our example we use a simplified substructure of a 380 kV energy transmission network.[1] Though specialized algorithms may be available for these networks, our main point is to illustrate our (domain independent) concepts and algorithms using a real-world example. We are currently working with several other types of systems, including both redundant and non-redundant systems and circuits.

In Figure 1 generators g_i supply the network with energy distributed by lines l_i and bus bars b_i. Transformers t_i feed the energy to consumers. Crossings are possible connections between lines and bus bars established by power switches. Closed power switches are represented by dots. We denote end points of lines by $l_i - b_j$. Some bus bars are doubled. Generator g_1 supplies the transformers t_1, t_2, and t_3 with energy. t_4 and t_5 are supplied by g_2.

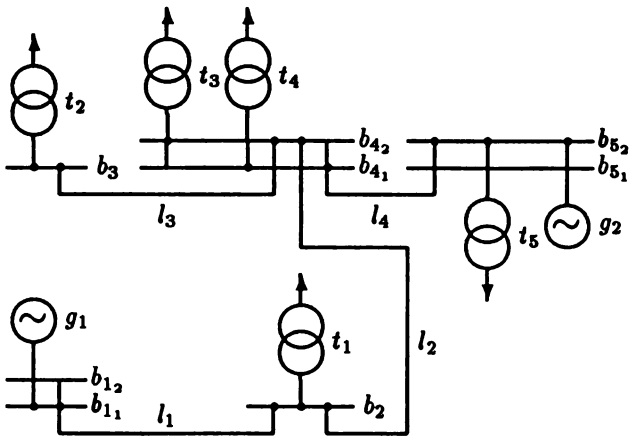


Figure 1: Energy Transmission Net

Common faults in such energy transmission networks are short circuits which cause serious damage of components if the faulty parts are not cut off in time. Therefore, a protection system monitors loads and trips the power switches if a dangerous overload is observed. Actions and measurements performed by the protection system are submitted to a central monitoring station. Voltages and loads of bus bars are measured every second. All data received (i.e. all actions and measurements performed by the system) are stored in a database, and a subset of them is subsequently used for fault localization.

Assume the following simplified fault scenario: Due to a high overload caused by a short circuit in a line the power switches at $l_1 - b_{1_1}$ and $l_3 - b_3$ are tripped. A short circuit at l_1 is half as likely as shorts at the other lines. Shorts at l_2 or l_3 are equally probable.

2.2 Separated Diagnosis/Repair Approach

Using a separate diagnosis/repair approach we analyze the data monitored. Considering the tripped switches we know that l_1, l_2, or l_3 are faulty, since a breaker is tripped only if energy flow during a short circuit is toward the line it is connected to. Additional information can be gained by observing the direction of the energy flow during the short circuit at breakers. In our example we consider two breakers $l_1 - b_2$ and $l_2 - b_{4_2}$ as possible places for probing. Since observing the energy flow at breaker $l_2 - b_{4_2}$ provides more information than observing it at $l_1 - b_2$ (using entropy based measurement selection), we choose $l_2 - b_{4_2}$. Using this information, we recognize that line l_1 or l_2 are faulty since the energy flow is towards l_2. The next measurement is the energy flow at breaker $l_1 - b_2$. Observing the energy flow towards l_1 yields l_1 as the only diagnosis.

After the diagnosis step a repair step analyzes the current status of the network recognizing that transformers t_1, t_2, and t_3 are off line. In order to re-establish their energy flow various switch actions have to be performed. Since bus bar b_{4_1} is supplied correctly with energy, transformer t_3 is moved from bus bar b_{4_2} to b_{4_1} (opening the switch at b_{4_2} and closing the switch at b_{4_1}). Similar, l_2 is moved to the correctly behaving bus bar after the switch at $l_1 - b_2$ had been opened and also line l_3 after closing $l_3 - b_3$.

2.3 Interleaved Diagnosis/Repair Approach

The conventional diagnosis approach tries to minimize the number of measurements and does not consider breakdown and repair costs. Since breakdown costs are enormous, re-establishing the correct behavior as soon as possible is the main goal. In our example the breakdown costs per time unit of the energy flow through transformer t_3 and t_1 are very high compared to the observation costs, while breakdown costs for t_2 are negligible.

In the described scenario full diagnosis identifying the most probable diagnosis is not needed in order to re-establish the energy flow through t_3. Using information about the tripped switches we suspect a short circuit either in l_1, l_2, or in l_3. This is sufficient for performing immediate repair actions, i.e. moving the transformer t_3 to another bus. Only then, a further diagnosis step is required to re-establish the energy flow through transformer t_1 and t_2. Since t_1 is much more important than t_2 we prefer those measurements which probably will provide us with enough information for re-establishing the energy flow through t_1. This is the observation of energy flow at $l_1 - b_2$. Applying the standard measurement selection function would prefer $l_2 - b_{4_2}$. Recognizing l_1 as the faulty component

[1] For a more thorough introduction to diagnosis strategies in these networks see [Beschta *et al.*, 1992].

gives us the necessary information to re-establish the energy flows through the remaining transformers. We have saved substantial breakdown costs by interleaving observations and actions.

3 Basic Concepts

3.1 Diagnosis and Repair

Let us first repeat some basic concepts defined in [Friedrich *et al.*, 1990], [Nejdl, 1991] and [Friedrich *et al.*, 1992]. We view *diagnosis* as the process of determining the actual state of a system. Before the diagnosis process, a diagnostic agent does not have enough information so the true state of affairs is only known to be among a finitely describable set of possibilities. The task of diagnosis is to determine which one of these possibilities — called *possible worlds* — corresponds to the actual world.

Further, the agent usually does not consider all possible worlds, but rather focuses on a subset of these worlds, called the set of *plausible worlds*. Most existing diagnosis systems, among them [de Kleer and Williams, 1989], [Struss and Dressler, 1989] and [Friedrich and Nejdl, 1990], already use such a focus to increase efficiency. Each plausible world represents one possibility an intelligent agent considers during the diagnosis and repair process.

Additionally, the goal of an extended *diagnosis and repair process* is to find and execute a general *diagnosis and repair plan*. This plan usually integrates observations and actions to (re-) establish certain properties of the system which are represented by the *system and repair purpose*. Furthermore, the plan should try to be optimal in the sense that it minimizes system breakdown costs as well as diagnosis and repair costs.

We use the term *observation procedures* for any actions that lead to new information about the actual state of affairs (including both simple measurements and complicated test procedures), while *action procedures* are used to change the actual state. *Plans* are partially ordered sets of actions. Each step in an *R-Plan* corresponds to one action procedure, whereas *DR-Plans* include both action and observation procedures. We then have the following definition of an extended *diagnosis and repair algorithm scheme*. In this paper we discuss the generation and evaluation of *DR-Plans* in more depth.

These ideas lead to the following definition of an extended *diagnosis and repair algorithm scheme* already discussed in [Friedrich *et al.*, 1990] and [Nejdl, 1991]. In this paper we will discuss the generation and evaluation of *DR-Plans* in more depth.

Algorithm Scheme 1 Characterization of *diagnosis and repair process*:

1. Start with an initial set of plausible worlds PW.
2. Take the purpose $\mathcal{T}$ representing the desired functionality after the repair process.
3. Generate DR-Plans consisting of observation/action procedures, evaluate their utility.
4. *If* best DR-Plan is empty (no advantageous observations/actions exist), *then* stop.
5. Execute the next procedure from the best DR-Plan, changing PW to PW'. *goto 3*

3.2 System Architecture

Plausible World Engine The first component of an MBR system, which we call a *plausible world engine*, represents the view of the system held by the diagnostic agent. It updates its knowledge in response to new information (made available by information procedures) and in response to actions executed by the agent changing the system or its environment (represented by action procedures). Its output is a set of plausible worlds representing the possibilities the agent is considering.

This plausible world engine is a generalization of diagnosis generation engines in current model-based diagnosis systems. It can compute not only the impact of new observations on the current view of the system, but also the impact of actions. Additionally while plausible worlds correspond in a sense to leading candidates in a diagnosis system they do not include only the mode information of components but represent the whole system state (either implicitly in form of a history like in [Friedrich *et al.*, 1992] or explicitly like in [Friedrich *et al.*, 1991, Friedrich and Lackinger, 1991]) which is necessary for computing the impact of observations and actions.

Plan/Selection Engine The second component of an MBR system, the *plan/selection engine*, represents the active planning part of the diagnostic agent. The possibilities it considers are represented by the set of plausible worlds. The component uses a set of available observation and action procedures, as well as a characterization of the system and repair purpose. Its task is to propose observation procedures to gain new information (restricting the set of plausible worlds) and action procedures transforming the system to comply with the system and repair purpose.

The plan/selection engines are rather restricted in current MBR systems as they consist only of measurement selection algorithms proposing observations to distinguish between the leading candidates (i.e. plausible worlds). Two additional extensions are proposed in this paper: Add a planning component generating repair plans for the plausible worlds, and propose both

observation and action procedures depending on which type of procedure is optimal in the current situation.

Acting and Sensing Given that planning in itself is a rather difficult endeavor, generating and evaluating plans from uncertain initial conditions seems to be too complex in general (compare e.g. [Chrisman and Simmons, 1991]). However, while this is true for optimal solutions, this paper shows that results can be achieved if we use a satisficing approach. We will use local optimization as much as possible, and as a result extend and improve current MBR measurement selection procedures based on one-step look-ahead strategies like entropy ([de Kleer and Williams, 1987]) while still maintaining their good computation properties. We will concentrate on the novel features of our approach and show how these can be integrated with a conventional planner. We will therefore assume the existence of a conventional planning module which is able to generate repair plans consisting only of action procedures (R-Plans) for a single world.

For complexity reasons, we will consider only the locally best plan for each world. If computation time is not critical, our algorithms can be changed easily to use more than one plan per world. On the other hand, if computation time is critical because of high breakdown costs, plans for probable faults may be precomputed and stored in a database. Another possibility is to store only emergency plans for important goals (i.e. in safety-critical systems) and to compute plans for the rest of the goals at runtime.

The remaining task is now to merge such a set of R-Plans (no matter how and when they are generated) together with observation (i.e. sensing) procedures in order to get improved diagnosis and repair plans (DR-Plans).

Diagnosis/Repair Plans If we use such a planning module together with the measurement selection module from [de Kleer and Williams, 1987] the diagnosis and repair plans (DR-Plans from now on) which will be generated have two basic steps. First, distinguish between all plausible worlds (leading candidates). Second, execute the R-Plan for the appropriate world.

The one step look-ahead entropy-based selection function which is used in GDE and described in [de Kleer and Williams, 1987] has proven very successful and is applied in many similar model-based diagnosis systems. It is nearly optimal in the sense that complete look-ahead leads to the same results in many cases ([de Kleer *et al.*, 1991]). However, this is valid only within the assumptions made in conventional diagnosis systems. If we take repair actions into account it is often advantageous (as in our introductory example) to cluster certain leading candidates and execute (part of) their R-Plans together before continuing to distinguish between them.

The DR-Plans generated by GDE and similar systems therefore are suboptimal in two respects. First, they may execute repair actions later than necessary, as they do not propose any repair actions until all candidates are distinguished. Second, they may choose sub-optimal measurements or make useless distinctions between candidates because they never consider clustering candidates based on their R-Plans.

This paper shows how to improve the selector module by recognizing advantageous clusters of worlds based on their R-Plans and thus optimizing measurements and by proposing both observations and actions based on utility-based evaluation of different DR-Plans.

The usual measurement selection module does not try to achieve optimal measurement but rather steers a middle ground between optimality of measurement selection and computational complexity. This will also be our guideline, which is even more needed in our case, as the problem of proposing observations *and* actions is inherently more complex than pure measurement selection.

We will start from the entropy-based measurement selection procedure in [de Kleer and Williams, 1987] and use the resulting diagnosis and repair plan as upper bound for cost minimization. Using a set of greedy algorithms, we will transform this plan into a plan which interleaves diagnosis and repair and is therefore able to better minimize the cost of the diagnosis and repair process.

4 Diagnosis/Repair Plans

4.1 Basic Framework

Assumptions For each plausible world $w_i \in PW$ we have one repair plan consisting only of action procedures. These plans (*R-Plans*) are provided by our conventional planning module. The plausible world is used as initial situation for the planner, a set of repair goals g_j as (partial) specification for the goal situation. Additionally the planner uses a set of available repair actions.

An R-Plan is a list of partially ordered sets of repair actions. We assume each serialization of such a set is possible, i.e. the partial order is a abbreviation of all these total orders. Additionally, we have a set of possible observation procedures obs_i that can be used to distinguish between the plausible worlds w_i.

Our task is now to take all these R-Plans plus additional observation procedures and integrate them into one DR-Plan which minimizes the sum of breakdown, repair, and diagnosis costs.

Structure of DR-Plans Before we can describe such a transformation procedure, we have to describe the structure of DR-Plans. The main unit of diagno-

sis in such a DR-Plan is no more the single world $w_i \in PW$ but rather a cluster $clu_k \subseteq PW$. We can distinguish between two alternating phases of the DR-Plan. First, distinguish between clusters of the current partitioning (*diagnosis phase*). Second, execute an optimal subset of plans in the relevant cluster (*repair phase*).

In each cluster, after having executed some action procedures, we again partition the remaining worlds in the cluster and proceed to distinguish between these clusters, etc. As these DR-Plans are only estimates for an optimal plan and are re-computed after new information is obtained (by observation procedures), we can simplify their structure by using only a restricted number of diagnosis/repair cycles to make the whole transformation process tractable. For our examples we use a two-step structure including four phases. First, distinguish between clusters of the first partitioning. Second, execute optimal subset of plans in the relevant cluster. Third, distinguish between the remaining worlds. Fourth, execute the remaining subset of plans for the current world.

In the next section we will discuss cost estimates for the general case, which can be easily simplified for n-step DR-Plan structures for a restricted n.

4.2 Cost Estimates of DR-Plans

During the time a specific purpose goal is not fulfilled, breakdown costs occur. These costs may increase over time and are therefore best estimated by an integral over time for this purpose goal. The breakdown costs for one cluster clu_k with respect to a set of purpose goals g_i ($c_{g_i}(t)$ being the breakdown costs for goal g_i per time unit) which are re-established in this cluster are therefore estimated by

$$c_{not(g)}(clu_k) = \sum_{g_i \in repaired(clu_k)} \int_{t_{now}}^{t_{repaired(g_i)}} c_{g_i}(t)dt$$

where $t_{repaired}(g_i)$ is estimated by

$$t_{repaired(g_i)} = t_{now} + \sum_{clu_j, clu_j < clu_k} \#\_\text{obs}(clu_j) \times t\_obs + \sum_{g_j, g_j < g_i} t_{rep(g_j)} + t_{rep(g_i)}$$

and the purposes g_i are re-established by a plan executed in this cluster.

t_obs is the average time to execute one observation procedure. $\#\_\text{obs}(clu_j)$ is the average number of measurements to distinguish between elements (estimated by the entropy between the clusters) of a cluster. Note, that the elements of a cluster are also clusters. A cluster may comprise all plausible worlds as well as only one world. $clu_j < clu_k$ means, that clu_k is a subcluster of clu_j. $g_j < g_i$ means, that the repair actions for g_j are done before the actions for g_i. $t_{rep(g_j)}$ is the time needed to execute the actions for g_j considering the actions already executed before to achieve earlier repaired goals.

The time point where we expect a goal g_i to be fulfilled is the current time point plus the time needed for all observation and action procedures until g_i is established. The expected costs of a partition *par* is the sum of expected breakdown, diagnosis, and repair costs:

$$c_{par}(par) = c_{not(g)}(par) + c_{dia}(par) + c_{rep}(par)$$

The expected breakdown, diagnosis, and repair costs are computed by using the following formulas. These costs are weighted sums of the costs of clusters where the weight is determined by the (conditional) probability of these clusters.

DR-Plan Breakdown Costs:

$$c_{not(g)}(par) = \sum_{clu_k \in par} p(clu_k) \times (c_{not(g)}(clu_k) + \sum_{clu_i \in clu_k} p(clu_i|clu_k) \times (c_{not(g)}(clu_i) + \ldots + \sum_{pw_l \in clu_j} p(pw_l|clu_j|\ldots|clu_i|clu_k) \times c_{not(g)}(pw_l)\ldots))$$

DR-Plan Diagnosis Costs:

$$c_{dia}(par) = c\_obs \times (\#\_obs(par) + \sum_{clu_k \in par} p(clu_k) \times (\#\_obs(clu_k) + \sum_{clu_i \in clu_k} p(clu_i|clu_k) \times (\#\_obs(clu_i) + \ldots)))$$

DR-Plan Repair Costs:

$$c_{rep}(par) = \sum_{clu_k \in par} p(clu_k) \times (c\_plan(clu_k) + \sum_{clu_i \in clu_k} p(clu_i|clu_k) \times (c\_plan(clu_i) + \ldots + \sum_{pw_l \in clu_j} p(pw_l|clu_j|\ldots|clu_i|clu_k) \times c\_plan(pw_l)\ldots))$$

where

$$p(clu_k) = \sum_{clu_j \in clu_k} p(clu_j)$$

c_obs are the average costs for one observation. $p(pw_l)$ are the probabilities of the different plausible worlds. $c\_plan(clu_k)$ is the sum of the costs of all actions executed in cluster clu_k. $c\_plan(pw_l)$ is the sum of the costs of all remaining actions necessary to guarantee the goals of world pw_l.

4.3 Generation of 2-Step DR-Plans (Partitions)

To generate a DR-Plan which is as optimal as possible without using too much computation time in the process we use the following greedy algorithm.

Algorithm 1 Generating and Evaluating 2-step DR-Plans:

```
function Generate_DR-Plan;
/* Start from GDE DR-Plan as upper bound,
where each world represents one cluster and
diagnosis therefore distinguishes
between each world. */
current_DR-Plan := GDE_DR-Plan;
loop forever
  /* Evaluate the merge of every pair of clusters
  by constructing the corresponding variation of the
  current DR-Plan and take the best one. */
  best_DR-Plan := current_DR-Plan;
  for i:=1 to number_of_clusters(current_DR-Plan) do
    for j:=i+1 to
    number_of_clusters(current_DR-Plan) do
      new_DR-Plan :=
        merge_clusters(current_DR-Plan,i,j);
      if cost(new_DR-Plan) < cost(best_DR-Plan)
      then
        best_DR-Plan := new_DR-Plan;
      end if
    end for
  end for
  /* If the best merge of two clusters is better than
  the current DR-Pan according to the cost function
  defined above, take this DR-Plan as the current
  one and loop again. */
  if cost(best_DR-Plan) < cost(current_DR-Plan)
  then
    current_DR-Plan := best_DR-Plan
  else
  /* else output the current DR-Plan as estimate
  for the best partitioning and exit the loop. */
    return current_DR-Plan;
    exit loop
  end if
end loop
end function
```

This algorithm has $O(n^3)$ basic steps, where n is the number of plausible worlds in PW.

This greedy algorithm represents of course a great improvement over an exact optimization algorithm, which would have to analyze all possible partitions. In mathematics their number is known as *Stirling Number of the Second Kind* and lies between 2^n and $n!$ ([Baron and Kirschenhofer, 1989]).[2]

While constructing the variation of a DR-Plan corresponding to a new cluster, we have to choose which actions in a cluster are done (in phase 2) before finally distinguishing between all worlds in the cluster (in phase 3). We use again a greedy algorithm for this task.

Algorithm 2 Construct Variation of DR-Plan:

```
function merge_clusters(original_DR-Plan,i,j)
current_DR-Plan :=
  transformation of original_DR-Plan,
  where the worlds of clusters i and j are merged
  into one cluster and phase 2 is set to empty
set_of_goals := all purpose goals;
loop forever
  /* for all remaining goals g_k do:
  Generate the (approximately optimal) merge of all
  sub-plans for this goal for all plausible worlds
  in the two clusters i,j. Compute the costs of
  the DR-Plan resulting from executing this merged
  plan in phase 2 */
  best_DR-Plan := current_DR-Plan;
  forall g_k ∈ set_of_goals do
    /* merge plans for g_k and include them at end
    of phase 2 */
    new_DR-Plan :=
      merge_plans(current_DR_Plan,g_k);
    /* Choose that DR-Plan, which minimizes
    costs. */
    if cost(new_DR-Plan) < cost(best_DR-Plan)
    then
      best_DR-Plan := new_DR-Plan;
      goal_to_be_merged := g_k
    end if
  end forall
  /* If this best DR-Plan has less costs than the
  current DR-Plan make this DR-Plan the current
  DR-Plan, reduce the goals considered by the goal
  moved to phase 2 and loop again. */
  if cost(best_DR-Plan) < cost(current_DR-Plan)
  then
    current_DR-Plan := best_DR-Plan
    set_of_goals := set_of_goals - goal_to_be_merged
  else
  /* else output the current DR-Plan as estimate for
  the best partitioning and exit the loop. */
    return current_DR-Plan;
    exit loop
  end if
end loop
end function
```

This algorithm has $O(g^2)$ basic steps, where g is the number of goals.

Our basic plan merging algorithm uses at most $O(n \times m)$ steps, where m is the maximal number of preconditions in a plan. The basic planning paradigm used is similar to STRIPS. We specify plans by purpose and action preconditions and by effects. This algorithm is sufficient for our example. However, we are also experimenting with more complex merging algorithms.

Overall Complexity To summarize, we have a polynomial approximation algorithm with an overall complexity of $O(n^3) \times O(g^2) \times O(n \times m)$ where n is the number of plausible worlds, g is the number of

[2] The exact Stirling Number of the Second Kind is $\sum_{k=1\ldots n} \frac{1}{k!} \sum_{j=0\ldots k} \binom{k}{j} (-1)^{k-j} j^n$

purpose goals and m is the maximal number of preconditions. We have therefore extended the measurement proposer of [de Kleer and Williams, 1987] to a measurement/action proposer while adding only a polynomial term to the overall complexity. Not included is the plan generation of R-Plans, which may however be done at compile time by storing (total or partial) R-Plans for the most important failures and goals.

5 Example Continued

Take again our example from Section 2. We assume that the failure probabilities for short circuits are very low so that it is sufficient to consider only single faults as plausible worlds. The probability of world l_1 (representing l_1 is faulty and all other components work correctly) is 0.2, the probability of world l_2 (and also of world l_3) is 0.4.

Table 1 shows our cost estimates including costs of observations and actions and the time needed to carry them out. The breakdown cost are stated in costs per time unit.

Actions	Costs	Time needed
$move(X)_T$	2	2
$close(X)_T$	1	1
$open(X)_T$	1	1
$repair(X)_T$	100	1000
Observations	**Costs**	**Time needed**
$observe(X)_T$	2	1

Purpose	BD/TU[a]
$energy(t1,+)_T$	100
$energy(t2,+)_T$	0.01
$energy(t3,+)_T$	1000

[a] Breakdown Costs per Time Unit

Table 1: Costs in Some Unit

Table 2 shows the repair actions for each plausible world and purpose we consider in the example[3]. For simplicity, we state these repair plans only partially, taking the current switch positions into account. The full plans include all switch positions such that the purpose is guaranteed.

In the following tables we use a notion like $\{\{l_1, l_2- < t_1, t_2 >\}, \{l_3- < t_1, t_2, t_3 >\}\}$ to indicate that the partition consists of clusters $\{l_1, l_2\}$ and $\{l_3\}$. In the first cluster the energy flow through transformer t_1 is re-established before that of t_2. All other purposes are re-established after diagnosis is done. The second

[3] We omit the temporal index. The temporal properties can be easily calculated using the number of actions and their duration

	$energy(t1,+)$	$energy(t2,+)$	$energy(t3,+)$
l_1	$open(l_1 - b_2)$ $move(l_2 - b_{4_1})$	$move(l_3 - b_{4_1})$ $close(l_3 - b_3)$	$move(t_3 - b_{4_1})$
l_2	$open(l_2 - b_2)$ $close(l_1 - b_{1_1})$	$move(l_3 - b_{4_1})$ $close(l_3 - b_3)$	$move(t_3 - b_{4_1})$
l_3	$open(l_3 - b_{4_2})$ $close(l_1 - b_{1_1})$	$open(l_3 - b_{4_2})$ $repair(l_3)$ $close(l_1 - b_{1_1})$ $close(l_3 - b_{4_2})$ $close(l_3 - b_3)$	$open(l_3 - b_{4_2})$ $close(l_1 - b_{1_1})$

Table 2: Repair Plans depending on Worlds and Purposes

Partition 1: $\{\{l_1- < t_3, t_1, t_2 >\}$,
$\{l_2- < t_3, t_1, t_2 >\}, \{l_3- < t_3, t_1, t_2 >\}\}$
Expected Costs: 4067.23
Partition 2: $\{\{l_1, l_2, l_3- < t_3 >\}\}$
Expected Costs: 2626.12

Table 3: Partitions and Their Expected Costs (Nothing Repaired):

cluster comprises only l_3. All three purposes are re-established in this cluster.

Table 3 shows two partitions and their expected costs. The DR-Plans for these partitions are shown in Table 4 and Table 5. The second partition where the purpose of t_3 is re-established with respect to all plausible worlds has significantly lower costs than all other possible partitions and is therefore preferred, i.e., t_3 is repaired before additional observations are considered. The costs are obtained by using our concepts for generating and evaluating DR-Plans.

Diagnosis: distinguish between $\{l_1\}, \{l_2\}, \{l_3\}$		
Re-est. t_3, t_1, t_2 in $\{l_1\}$	Re-est. t_3, t_1, t_2 in $\{l_2\}$	Re-est. t_3, t_1, t_2 in $\{l_3\}$
$move(t_3 - b_{4_1})$	$move(t_3 - b_{4_1})$	$open(l_3 - b_{4_2})$
/* t_3 on line */	/* t_3 on line */	$close(l_1 - b_{1_1})$
$open(l_1 - b_2)$	$open(l_2 - b_2)$	/* t_3 on line */
$move(l_2 - b_{4_1})$	$close(l_1 - b_{1_1})$	/* t_1 on line */
/* t_1 on line */	/* t_1 on line */	$repair(l_3)$
$move(l_3 - b_{4_1})$	$move(l_3 - b_{4_1})$	$close(l_3 - b_{4_2})$
$close(l_3 - b_3)$	$close(l_3 - b_3)$	$close(l_3 - b_3)$
/* t_2 on line */	/* t_2 on line */	/* t_2 on line */

Table 4: DR-Plan for Partition 1

After re-establishing the energy flow through t_3, partitions are again evaluated. Table 6 shows the expected costs of two alternative partitions, Table 7 and Table 8 show their DR-Plans. The partition using $\{l_2, l_3\}$ and $\{l_1\}$ as clusters has significantly lower costs. Consequently, we use this partition to guide the measurement selection and therefore prefer observing the energy flow at $l_1 - b_2$ since $l_2 - b_{4_2}$ does not discri-

Re-est. t_3 in $\{l_1, l_2, l_3\}$		
$move(t_3 - b_{4_1})$ /* t_3 on line */		
Diagnosis: distinguish between $\{l_1\}, \{l_2\}, \{l_3\}$		
Re-est. t_1, t_2 in $\{l_1\}$	Re-est. t_1, t_2 in $\{l_2\}$	Re-est. t_1, t_2 in $\{l_3\}$
$open(l_1 - b_2)$ $move(l_2 - b_{4_1})$ /* t_1 on line */ $move(l_3 - b_{4_1})$ $close(l_3 - b_3)$ /* t_2 on line */	$open(l_2 - b_2)$ $close(l_1 - b_{1_1})$ /* t_1 on line */ $move(l_3 - b_{4_1})$ $close(l_3 - b_3)$ /* t_2 on line */	$open(l_3 - b_{4_2})$ $close(l_1 - b_{1_1})$ /* t_1 on line */ $repair(l_3)$ $close(l_3 - b_{4_2})$ $close(l_3 - b_3)$ /* t_2 on line */

Table 5: DR-Plan for Partition 2

minate. If we used Partition 2.1 then measurement selection would chose $l_2 - b_{4_2}$ as observation point. In Partition 2.2, the action $close(l_2 - b_2)$ (marked with $\star$) is an example for an action, which was not necessary in the original state, but has become necessary after the repair actions done prior to this action.

Partition 2.1: $\{\{l_1 - < t_1, t_2 >\}, \{l_2 - < t_1, t_2 >\}, \{l_3 - < t_1, t_2 >\}\}$
Expected Costs: 424.10

Partition 2.2: $\{\{l_2, l_3 - < t_1 >\}, \{l_1 - < t_1, t_2 >\}\}$
Expected Costs: 344.91

Table 6: Partitions and Their Expected Costs (t_3 Repaired)

Diagnosis: distinguish between $\{l_1\}, \{l_2\}, \{l_3\}$		
Re-est. t_1, t_2 in $\{l_1\}$	Re-est. t_1, t_2 in $\{l_2\}$	Re-est. t_1, t_2 in $\{l_3\}$
$open(l_1 - b_2)$ $move(l_2 - b_{4_1})$ /* t_1 on line */ $move(l_3 - b_{4_1})$ $close(l_3 - b_3)$ /* t_2 on line */	$open(l_2 - b_2)$ $close(l_1 - b_{1_1})$ /* t_1 on line */ $move(l_3 - b_{4_1})$ $close(l_3 - b_3)$ /* t_2 on line */	$open(l_3 - b_{4_2})$ $close(l_1 - b_{1_1})$ /* t_1 on line */ $repair(l_3)$ $close(l_3 - b_{4_2})$ $close(l_3 - b_3)$ /* t_2 on line */

Table 7: DR-Plan for Partition 2.1

Table 9 shows the actual total costs of our scenario comparing the conventional separated diagnosis/repair approach with our method for each plausible world. The last column represents the weighted sum with respect to the normalized probabilities of the three plausible worlds. Table 10 shows the sequence of observations and actions actually performed (in the case where l_1 is faulty) by systems using an interleaved and a separated diagnosis/repair approach, respectively.

In all three plausible failure scenarios our diagnosis/repair method outperforms the separated diagnosis/repair approach. Cost estimation is close to the

Diagnosis: distinguish between $\{l_1\}, \{l_2, l_3\}$	
Re-est. t_1, t_2 in $\{l_1\}$	Re-est. t_1 in $\{l_2, l_3\}$
$open(l_1 - b_2)$ $move(l_2 - b_{4_1})$ /* t_1 on line */ $move(l_3 - b_{4_1})$ $close(l_3 - b_3)$ /* t_2 on line */	$open(l_2 - b_2)$ $close(l_1 - b_{1_1})$ /* t_1 on line */
	Diagnosis: distinguish between $\{l_2\}, \{l_3\}$
	Re-est. t_2 in $\{l_2\}$ / Re-est. t_2 in $\{l_3\}$
	$move(l_3 - b_{4_1})$ $close(l_3 - b_3)$ /* t_2 on line */ / $open(l_3 - b_{4_2})$ $repair(l_3)$ $close(l_2 - b_2)$ $\star$ $close(l_3 - b_{4_2})$ $close(l_3 - b_3)$ /* t_2 on line */

Table 8: DR-Plan for Partition 2.2

Approach	World l_1	World l_2
Separated	4712.09	4611.09
Interleaved	2610.09	2511.09
Approach	World l_3	$\emptyset$
Separated	3416.05	4153.27
Interleaved	2620.08	2574.49

Table 9: Diagnosis Repair Approaches and Their Actual Total Costs

weighted sum of the actual costs.

6 Comparison

Previous work in our group has mainly dealt with the logical definition of the concepts involved in the diagnosis and repair process and its general framework ([Friedrich *et al.*, 1990], [Friedrich *et al.*, 1991], [Friedrich *et al.*, 1992], [Nejdl, 1991]). In [Freitag and Friedrich, 1992] this framework was used to develop structural focusing techniques exploiting a specified system purpose. The current paper can be seen as a detailed specification of utility evaluation and selection of actions/observations defined in our framework.

Other interesting papers are the ones by Crow and Rushby ([Crow and Rushby, 1991]) and by Poole and Provan ([Poole and Provan, 1991], [Provan and Poole, 1991]), though they use a much more abstract notion of utility. These papers lack the crucial notion of system or repair purpose and are therefore not able to discuss purpose dependent repairs. They also include all appropriate repair plans (reconfigurations in [Crow and Rushby, 1991] and treatments in [Poole and Provan, 1991]) into their model which is not feasible in the general case.

An interesting notion in [Poole and Provan, 1991] is the notion of appropriate actions. In order to iden-

Interleaved	Separated
$move(t_3 - b_{4_1})$	observe at $l_2 - b_{4_2}$
/* t_1 on line */	observe at $l1 - b2$
observe at $l1 - b2$	/* faulty line
/* faulty line l_1	identified */ l_1
identified */	$move(t_3 - b_{4_1})$
$open(l_1 - b_2)$	/* t_3 on line */
$move(l_2 - b_{4_1})$	$open(l_1 - b_2)$
/* t_1 on line */	$move(l_2 - b_{4_1})$
$move(l_3 - b_{4_1})$	/* t_1 on line */
$close(l_3 - b_3)$	$move(l_3 - b_{4_1})$
/* t_2 on line */	$close(l_3 - b_3)$
	/* t_2 on line */

Table 10: Actual Sequence of Observations and Actions

tify these actions Poole and Provan basically take all available repair actions and use them (implicitly) to define static partitions such that the same set of repair actions can be applied to all diagnoses in a given cluster. Diagnosis is then done only up to these partitions. Such an approach still finishes diagnosis before repair and therefore cannot be applied in our example, where the different diagnoses considered share some repair actions but not all.

A recent paper by Chen and Srihari [Chen and Srihari, 1992] describes the selection of observations and replacements of components. No other repair actions are considered and the utility function does not include breakdown costs and repair goals.

Our notion of plausible worlds is a generalization of the leading candidates of de Kleer and Williams ([de Kleer, 1991]). Compared to a candidate (or diagnosis), a world not only specifies the defect components but the whole system (providing a complete initial state of affairs for the planning step).

7 Conclusions

This paper presents a detailed specification of the utility evaluation of diagnosis/repair plans continuing previous work of our group in this area. Our approach results in a reduction of overall diagnosis/repair costs by interleaving repair actions and observations. We extend the current model of measurement selection used in MBD systems by a decision theoretic model usable for a diagnosis/repair process. Our system dynamically proposes both actions and observations using utility functions based on time-dependent breakdown costs and observation/repair costs. These are defined using the specific notion of system and repair purposes. Repair actions include arbitrary actions as specified in a set of available repair actions. Observation procedures include both ordinary measurements and complex test actions.

Using a satisficing approach we are able to transform (in poly-time) a set of conventional plans (consisting only of actions) appropriate for a set of plausible descriptions of the current state into one enhanced plan including additional sensing operations appropriate for the current state as a whole. We are able to retain the complexity of conventional planners working only with actions in our planning phase and the complexity of conventional measurement selection procedures which do not take any repair actions into account.

8 Acknowledgement

We thank Susanne Albinger for implementing parts of the plan/selection engine.

References

[Baron and Kirschenhofer, 1989] Gerd Baron and Peter Kirschenhofer. *Einführung in die Mathematik für Informatiker*. Springer-Verlag, 1989.

[Beschta *et al.*, 1992] Anton Beschta, Oskar Dressler, Hartmut Freitag, Michael Montag, and Peter Struss. A model-based approach to fault localization in power transmission networks. *Intelligent Systems Engineering*, 1992.

[Chen and Srihari, 1992] Jiah-shing Chen and Sargur N. Srihari. Action selection in interactive model-based diagnosis. In *Proceedings of the IEEE Conference on Artificial Intelligence Applications (CAIA)*, pages 67–73, Monterey CA, March 1992.

[Chrisman and Simmons, 1991] Lonnie Chrisman and Reid Simmons. Sensible planning: Focusing perceptual attention. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 756–761, Los Angeles, July 1991. Morgan Kaufmann Publishers, Inc.

[Crow and Rushby, 1991] Judith Crow and John Rushby. Model-based reconfiguration: Toward an integration with diagnosis. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 836–841, Los Angeles, July 1991. Morgan Kaufmann Publishers, Inc.

[de Kleer and Williams, 1987] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[de Kleer and Williams, 1989] Johan de Kleer and Brian C. Williams. Diagnosis with behavioral modes. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1324–1330, Detroit, August 1989. Morgan Kaufmann Publishers, Inc.

[de Kleer *et al.*, 1991] Johan de Kleer, Olivier Raiman, and Mark Shirley. One step lookahead is pretty good. In *Second International Workshop on*

the Principles of Diagnosis, Milano, Italy, October 1991.

[de Kleer, 1991] Johan de Kleer. Focusing on probable diagnoses. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 842–848, Anaheim, July 1991. Morgan Kaufmann Publishers, Inc.

[Freitag and Friedrich, 1992] Hartmut Freitag and Gerhard Friedrich. Focusing on independent diagnosis problems. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA, October 1992. To appear.

[Friedrich and Lackinger, 1991] Gerhard Friedrich and Franz Lackinger. Diagnosing temporal misbehavior. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1116–1122, Sydney, Australia, August 1991. Morgan Kaufmann Publishers, Inc.

[Friedrich and Nejdl, 1990] Gerhard Friedrich and Wolfgang Nejdl. MOMO — Model-based diagnosis for everybody. In *Proceedings of the IEEE Conference on Artificial Intelligence Applications (CAIA)*, Santa Barbara, March 1990. A slightly revised and extended version appears in Readings in Model-Based Diagnosis (Morgan Kaufmann, 1992).

[Friedrich *et al.*, 1990] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. Hypothesis classification, abductive diagnosis and therapy. In *Proceedings of the International Workshop on Expert Systems in Engineering*, Vienna, September 1990. Springer Verlag, Lecture Notes in Artificial Intelligence, Vo. 462.

[Friedrich *et al.*, 1991] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. Towards a theory of the repair process. In *Proceedings of the Portuguese Conference on Artificial Intelligence*, Albufeira, October 1991. Springer AI Lecture Notes. Also appeared at the Model-Based Reasoning Workshop, AAAI'91, July 1991, Anaheim.

[Friedrich *et al.*, 1992] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. Formalizing the repair process. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, Vienna, August 1992. Also appeared in the Proceedings of the Second International Workshop on Principles of Diagnosis, Milano, 1991.

[Nejdl, 1991] Wolfgang Nejdl. Belief revision, diagnosis and repair. In *Proceedings of the International GI Conference on Knowledge Based Systems*, München, October 1991. Springer-Verlag.

[Poole and Provan, 1991] David Poole and Gregory M. Provan. Use and granularity in consistency-based diagnosis. In *Proceedings of the Second International Workshop on Principles of Diagnosis*, Milano, September 1991.

[Provan and Poole, 1991] Gregory M. Provan and David Poole. The utility of consistency-based diagnostic techniques. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 461–472, Cambridge, MA, April 1991. Morgan Kaufmann Publishers, Inc.

[Struss and Dressler, 1989] Peter Struss and Oskar Dressler. Physical negation — Integrating fault models into the general diagnostic engine. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1318–1323, Detroit, August 1989. Morgan Kaufmann Publishers, Inc.

Abductive Plan Recognition and Diagnosis: A Comprehensive Empirical Evaluation

Hwee Tou Ng and **Raymond J. Mooney**
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
htng@cs.utexas.edu, mooney@cs.utexas.edu

Abstract

While it has been realized for quite some time within AI that abduction is a general model of explanation for a variety of tasks, there have been no empirical investigations into the practical feasibility of a general, logic-based abductive approach to explanation. In this paper we present extensive empirical results on applying a general abductive system, ACCEL, to moderately complex problems in plan recognition and diagnosis. In plan recognition, ACCEL has been tested on 50 short narrative texts, inferring characters' plans from actions described in a text. In medical diagnosis, ACCEL has diagnosed 50 real-world patient cases involving brain damage due to stroke (previously addressed by set-covering methods). ACCEL also uses abduction to accomplish model-based diagnosis of logic circuits (a full adder) and continuous dynamic systems (a temperature controller and the water balance system of the human kidney). The results indicate that general purpose abduction is an effective and efficient mechanism for solving problems in plan recognition and diagnosis.

1 INTRODUCTION

Finding explanations for events and actions is an important aspect of general intelligent behavior. A diverse set of intelligent activities, including natural language understanding, diagnosis, scientific theory formation, and image interpretation, requires the ability to construct explanations for observed phenomena. For instance, in text understanding, a reader infers the high-level goals and plans of the characters in a text in order to explain the events and actions described in the text. Such inference is called *plan recognition*, which is known to be an important component of text understanding [Allen, 1987]. Similarly, in medical diagnosis, based on the observed symptoms of a patient, a physician infers the possible diseases that may explain the symptoms. In physical device diagnosis, based on the observed misbehavior of a physical device, a diagnostician infers the possible faults that may explain the misbehavior.

In this paper, we view explanation as *abduction*. The standard logical definition of abduction is: given a set of axioms T (the domain theory) and a conjunction of atoms O (the observations), find minimal sets of atoms A (the assumptions) such that $A \cup T \models O$ and $A \cup T$ is consistent [Charniak and McDermott, 1985; Levesque, 1989].[1]

While it has been realized for quite some time within AI that abduction is a general model for explanation [Charniak and McDermott, 1985], there have been no empirical investigations into the practical feasibility of such a general abductive approach to explanation. Many important questions remain unexplored. For example, is it possible to have a general-purpose yet efficient algorithm that can be used for making useful abductive inference and solving moderately complex problems in reasonable time in all the various domains? Do we need special-purpose control heuristics separately tailored for each domain to achieve efficient abduction? Do the criteria for selecting the best explanations vary according to the domain? How difficult is it to encode the knowledge necessary for constructing explanations in the various domains?

This paper attempts to address these important issues. Towards this end, we have built a domain-independent system called ACCEL (Abductive Construction of Causal Explanations in Logic). In our system, knowledge about a variety of domains is uniformly encoded in first-order Horn-clause axioms. A general-purpose abduction algorithm, AAA (ATMS-based Abduction Algorithm), efficiently constructs explanations in these

[1] In this paper, the terms "abduction" and "abductive" are used to refer to this specific logical formulation as opposed to the more general notion of any method for inferring cause from effect.

domains. We have applied our abductive system to two general tasks: plan recognition in text understanding, and diagnosis of medical diseases, logic circuits, and dynamic systems. In plan recognition, ACCEL has been tested on 50 narrative texts, where each text consists of 1–4 short sentences. The system infers the high-level plans of characters based on the actions described in a text. In medical diagnosis, ACCEL diagnoses 50 real-world patient cases using a sizable knowledge base with over six hundred symptom-disease rules. The disorders are damaged regions of a human brain due to stroke. These cases have been previously used to test set covering methods [Tuhrim *et al.*, 1991; Peng and Reggia, 1990]. ACCEL also achieves model-based diagnosis via abduction, and has successfully diagnosed a full adder (an example of discrete, combinational logic circuits), a temperature controller, and the water balance system of the human kidney (examples of continuous dynamic systems).

The rest of this paper is organized as follows. Section 2 gives a brief overview of ACCEL. Section 3, 4, and 5 present empirical results for plan recognition, set-covering-based diagnosis, and model-based diagnosis, respectively. Section 6 discusses the results and presents the conclusions.

2 OVERVIEW OF ACCEL

Given an existentially quantified conjunction of atoms that encodes the input observations, and a set of first-order Horn clauses that encodes the domain theory, the algorithm AAA computes abductive explanations by backward-chaining on the observations, much like Prolog. However, even when there is no fact or consequent of a rule that unifies with a subgoal in the current proof attempt, instead of failing, the algorithm has the choice of making the subgoal an assumption if it is consistent to do so. The requirement for consistency means that abduction is in general undecidable. In ACCEL, inconsistency is detected using a predetermined list of *nogoods*, and by procedural code (for efficiency reasons). A nogood is a set of assumptions that implies falsity, and consistency checking ensures that an assumed set of atoms is not subsumed by any nogoods.

ACCEL can be used to compute all minimal sets of abductive assumptions; however, minimality is generally too unrestrictive and even in the propositional case, the number of minimal explanations can grow exponentially [Selman and Levesque, 1990]. Consequently, beam-search is generally used to limit computation to a fixed-sized subset of the currently best partial explanations according to a user-defined evaluation function. In a further attempt to improve efficiency, ACCEL performs ATMS-like caching of partial explanations. Empirical results have shown that caching can achieve more than an order of magnitude speedup in run time. More details on ACCEL and the AAA algorithm are given in [Ng, 1992]. (A previous version of ACCEL is described in [Ng and Mooney, 1991].)

3 ABDUCTIVE PLAN RECOGNITION

Given a logical representation of the literal meaning of a narrative text in terms of an existentially quantified conjunction of input atoms, ACCEL infers an "embellished" interpretation by constructing an abductive proof in which a set of higher-level plans is assumed and the assumed plans logically entail the characters' observed actions. An abductive proof is considered an interpretation of the input sentences. We do not focus on the parsing aspect of natural language understanding, and ACCEL does not accept natural language input.

Examples of the 50 narrative texts processed by ACCEL include: "Bill went to the liquor-store. He pointed a gun at the owner."; "Bill took a bus to a restaurant. He drank a milkshake. He pointed a gun at the owner. He got some money from him."; "Fred got a gun. He went to the restaurant. He packed a suitcase."; etc. The knowledge base axioms are formulated such that higher-level plans (like shopping and robbing) together with appropriate role-filler assumptions (like someone is the shopper of a shopping plan or the robber of a robbing plan) imply the input atoms representing the observed actions (like going to a store and pointing a gun).

In the plan recognition domain, explanations are evaluated by a criterion called *explanatory coherence* [Ng and Mooney, 1990; Ng, 1992]. This criterion attempts to capture the notion that natural language text is coherently structured and therefore explanations that better "tie together" various parts of the input sentences are to be preferred. Hence, evaluating explanations based on explanatory coherence takes into account the well known "Grice's conversational maxims" [Grice, 1975], which are principles governing the production of natural language utterances, such as "be relevant", "be informative", etc.

Specifically, the coherence metric C is defined as follows:

$$C = \frac{\sum_{1 \le i < j \le l} N_{i,j}}{l(l-1)/2}$$

where l = the total number of input atoms; and $N_{i,j}$ = 1 if there is some node n in the proof graph such that there is a (possibly empty) sequence of directed edges from n to n_i and a (possibly empty) sequence of directed edges from n to n_j, where n_i and n_j are input atoms. Otherwise, $N_{i,j} = 0$.[2] More details of

[2]The definition of coherence given in this paper is a

the coherence metric are described in [Ng and Mooney, 1990; Ng, 1992].

In the domain of plan recognition, we have tested ACCEL on 50 short narrative texts. To facilitate comparison between different approaches, the first 25 texts were taken from Goldman's PhD thesis [Goldman, 1990], where they were used to test a probabilistic approach to text understanding. An additional set of 25 similar narrative texts were created by Ray Mooney unbeknown to the system developer (Hwee Tou Ng). The intent is that the additional 25 examples will test for other novel combinations and sequences of actions that the knowledge base constructed for the initial 25 examples in principle should be able to handle. We will call the first set of 25 examples the *training* examples, and the second set of 25 examples the *test* examples.

The plans in the knowledge base include shopping, robbing, restaurant dining, traveling in a vehicle (bus, taxi, or plane), partying, and jogging. Each of these plans in turn has subplans, and some of the plans contain recursive subplans. For instance, traveling by plane includes the subplan of traveling (in some vehicle) to the airport to catch a plane. For each example, a set of input atoms representing the sentences is given to ACCEL. To give a sense of the size of our examples and the knowledge base used, there is a total of 107 KB rules and 70 taxonomy-sort symbols. Every taxonomy-sort symbol p will add an axiom (in addition to the 107 KB rules) of the form $inst(X, p) \rightarrow inst(X, supersort\text{-}of\text{-}p)$. The average number and maximum number of input atoms per example are 12.6 and 26 respectively. The knowledge base and the 50 examples are included in [Ng, 1992].

For each example, the correct explanation was determined based on the authors' intuition before running the example. To measure the quality of an explanation computed by ACCEL, we compared it to the correct explanation and recorded three error rates: the recall error rate R = the number of missing assumptions divided by the number of assumptions in the correct explanation, the precision error rate P = the number of excess assumptions divided by the number of assumptions in the computed explanation, and the overall error rate O = the average of the recall and precision error rates. (We used similar quality measures and terminology as in [Lehnert and Sundheim, 1991].) If more than one best explanations are computed for an example, we take the error rates for the example to be the average of the error rates over all the best explanations.

We ran ACCEL on the 50 examples using two different evaluation metrics: the coherence metric (breaking ties based on simplicity, defined as the number of assumptions made in an explanation) and the simplicity metric. The empirical results are summarized in Table 1, which shows the average recall (R), precision (P), and overall (O) error rates for the training examples, test examples, and all examples. The average run time per example is 1.83 minutes on a Sun Sparc 2 workstation.

Table 1: Empirical Results Comparing Coherence and Simplicity.

Example type	Coherence			Simplicity		
	R	P	O	R	P	O
Training	0.2%	0%	0.1%	26%	25%	25%
Test	2%	2%	2%	39%	38%	38%
All	1.1%	1%	1%	32%	31%	32%

The empirical results demonstrate that ACCEL can efficiently process these narrative texts, and it is sufficiently general to be able to handle similar plan recognition problems not known to the system developer in advance. Furthermore, coherence consistently performs better than simplicity on the examples tested.

Even though our knowledge base does not contain any probabilistic or likelihood information, the results on the training examples achieved by ACCEL are the same as those of Goldman' system which uses a probabilistic approach to language understanding. In the probabilistic approach, the primary purpose of *a priori* probabilities is to select a most likely explanation when there are otherwise multiple competing explanations. In ACCEL, we accomplish an analogous effect by writing axioms that only explain some input atom in terms of a high-level plan but not the other competing plans. More details are given in [Ng, 1992].

4 DIAGNOSIS BASED ON SET COVERING

Over the past decade, Reggia and his colleagues have developed an increasingly sophisticated theory of diagnosis based on set covering and applied the theory primarily to medical disease diagnosis [Peng and Reggia, 1990]. The basic diagnostic problem in the Generalized Set Covering (GSC) model is defined by four sets, (D, M, C, M^+), where D is a finite set of potential disorders, M is a finite set of potential manifestations (symptoms), $C \subseteq D \times M$ is a causation relation where $(d, m) \in C$ means "d may cause m", and $M^+ \subseteq M$ is the set of observed manifestations for the current case. $E \subseteq D$ is called a *cover* of M^+ iff for each $m \in M^+$ there exists $d \in E$ such that $(d, m) \in C$. A cover is said to be *minimum* if its cardinality is the smallest among all covers and *irredundant* (*minimal*) if none of its proper subsets is also a cover. Depending on the

slight modification of the one given in [Ng and Mooney, 1990]. The new definition remedies the anomaly reported in [Norvig and Wilensky, 1990] of occasionally preferring spurious interpretations of greater depths.

domain, one may consider all minimum or all minimal covers of the observed symptoms as the best diagnoses.

We can map a GSC diagnostic problem into an abduction problem in ACCEL as follows: Let the domain theory T be the set of axioms $\{d \rightarrow m | (d, m) \in C\}$, and let the input atoms $O = \bigwedge_{m \in M+} m$. Suppose only atoms $d \in D$ are assumable (i.e., we use predicate specific abduction [Stickel, 1988] in which only atoms with certain predicates are assumable). It can be easily proved that the set of covers of GSC is the same as the set of explanations in ACCEL [Ng, 1992].[3]

The logical abduction approach, being based on a more expressive representation language, can accommodate more naturally "causal chaining" [Peng and Reggia, 1990], incompatible disorders, and symptoms caused by combinations of disorders. As GSC diagnostic problems can be nicely represented as abduction problems, the remaining question is whether a general logic-based abductive system can solve such problems efficiently. Further, because the GSC diagnostic problem is NP-hard [Reggia *et al.*, 1985], the issue then becomes whether a logical abductive system can solve real problems in reasonable time and is competitive with existing set-covering algorithms. To address this issue, we tested ACCEL on the medical problem studied in [Tuhrim *et al.*, 1991], which specifies 25 brain areas (e.g. right frontal lobe) whose damage can explain 37 basic symptom types (e.g. impaired gag reflex). The knowledge base is quite large, consisting of 648 rules of the form $d \rightarrow m$. We were only able to obtain 50 of the original 100 cases from the authors of the initial study, each consisting of an average of 8.56 symptoms.

ACCEL efficiently computed all of the minimal (w.r.t. subset) explanations in an average of 2.4 seconds per case on a Sun Sparc 2 workstation. Unfortunately, we could not compare this result to that obtained in the original study, since no information on run time was provided. However, the empirical results strongly suggest that a general abductive system can solve real diagnostic problems in reasonable time.

Since abduction computes the same explanations as set covering when given the same evaluation criteria, ACCEL should replicate the accuracy results of the original study. As discussed in the original study, minimality is too unrestrictive to produce useful results (ACCEL returned an average of 26.6 minimal diagnoses per case). With minimum cardinality, ACCEL produced an average of only 4.6 diagnoses per case. In 44% of the cases, one of these diagnoses matches the expert's exactly; and in another 46% of the cases, one of the system's diagnoses was a subset or superset of the expert's (called a "close match" in [Tuhrim *et al.*, 1991]). The remaining 10% of the cases have a diagnosis that either partially matches the expert's (2%) or all of the diagnoses are totally wrong (8%). These results are slightly better than those reported in the original study: 6.5 diagnoses/case with 40% exact, 38% close, 5% partial, 17% wrong. This is presumably due to the fact that our results are based on only 50 of the original 100 cases. Two other evaluation metrics reported in the original study, most-probable and minimum-collapsed, performed even better. In [Tuhrim *et al.*, 1991], it is claimed that, although there have been no direct comparisons, the results from any of the covering metrics appear more promising than those obtained from standard rule-based approaches to this problem.

5 MODEL-BASED DIAGNOSIS VIA ABDUCTION

ACCEL also performs model-based diagnosis, which concerns inferring faults from first principles given knowledge about the correct structure and behavior of a system. Much research in model-based diagnosis has taken the *consistency-based* approach and has been applied primarily to devices with static, persistent states such as combinational logic circuits [Davis, 1984; de Kleer and Williams, 1987; Reiter, 1987; de Kleer and Williams, 1989]. In the consistency-based approach, a diagnosis is a set of normality and abnormality assumptions about device components that are *consistent* with the observations and the system description. This is in contrast to the *abductive* approach of diagnosis used in ACCEL, where normality and abnormality assumptions about device components together with the system description must *imply* or *explain* the observations.

Poole has proved that the consistency-based and abductive approaches are equivalent for propositional theories [Poole, 1988], and Konolige has extended the conditions under which equivalence holds to general first-order causal theories allowing for correlations, uncertainty, and acyclicity in the causal structure [Konolige, 1992].[4] In view of such formal equivalence results, issues such as ease of representation and computational efficiency are most important. Our empirical results suggest that a number of diagnostic problems, ranging from combinational logic circuits to continuous dynamic systems such as a proportional temperature controller and the water balance system of the human kidney, can be effectively represented and efficiently diagnosed using an abductive approach.

Research in model-based diagnosis can also be clas-

[3] Actually, that all minimal covers of GSC are all minimal explanations in abduction also follows as a corollary of two published theorems, Theorem 7.1 in [Reiter, 1987] and Theorem 4.2 in [Poole, 1988].

[4] Abduction appears to be better in some cases, as Konolige has reported that "the utility of the consistency based method is open to question", since in explanatory diagnostic tasks, "the answers it produces may have elements that are not relevant to a causal explanation" [Konolige, 1992, page 257].

sified according to whether information about fault models is utilized in diagnosis. The ***normality-based*** approach of [Reiter, 1987; de Kleer and Williams, 1987] does not utilize fault models and any misbehavior differing from the correct functioning of a device can be diagnosed. However, the lack of fault models may result in hypothesizing implausible faults [de Kleer and Williams, 1989; Struss and Dressler, 1989]. On the other hand, the work of [Dvorak and Kuipers, 1989] is ***fault-based*** in that the fault models are *a priori* determined and given to the diagnostic system. Hence, unanticipated faults are not detected. ACCEL combines both normality-based and fault-based diagnosis in that information about fault models is used in diagnosis and any deviation from the correct behavior can be diagnosed. The diagnostic systems Sherlock [de Kleer and Williams, 1989] and GDE+ [Struss and Dressler, 1989] have similar capability.

In the model-based diagnosis domain, ACCEL uses predicate specific abduction, where the assumable atoms include component behavioral mode assumptions of three types: (1) a component is normal; (2) a component is in some known fault mode; or (3) a component is abnormal (but not necessarily in any known fault mode). Other assumable atoms are "auxiliary" assumptions including assumptions that the input values of a device are as given, and in dynamic system diagnosis, that some qualitative magnitude is positive/negative, that two qualitative values obey some corresponding value constraint, etc. (More details about these auxiliary assumptions will be provided later.) Explanations in this domain are evaluated based on simplicity, where the best explanation is one with the least number of components that are not normal, which include components that are in some known fault mode and those that are not. Normality assumptions and auxiliary assumptions are "free" and do not affect the simplicity metric of an explanation. If two explanations have the same number of components that are not normal, then the one with the most number of components that are in some known fault mode is preferred.

5.1 DIAGNOSING LOGIC CIRCUITS

In this section, we describe how the abductive approach of ACCEL is used to diagnose a full adder which is representative of standard, combinational logic circuits. Figure 1 shows a full adder which consists of 2 exclusive-or gates (x1, x2), two and gates (a1, a2), and one or gate (o1). We assume that each gate has 4 behavioral modes: normal (the output bit reflects the correct gate behavior at all times), stuck-at-0 (the output bit is stuck at 0 regardless of the input bits), stuck-at-1 (the output bit is stuck at 1 regardless of the input bits), and abnormal (the behavior of the gate is unconstrained).

The knowledge base axiom that describes the correct

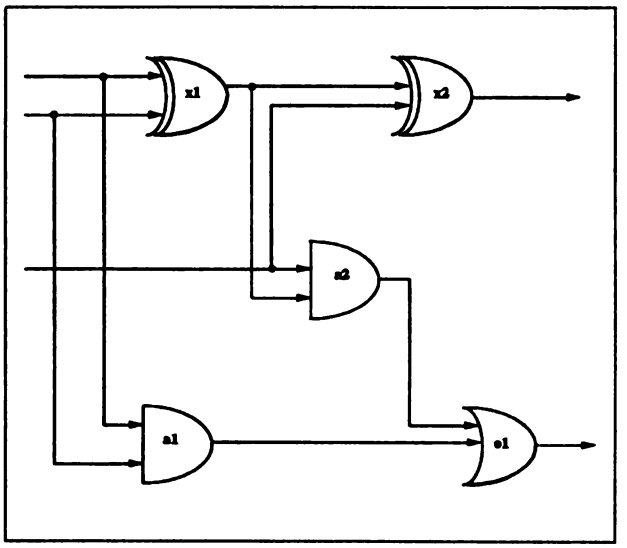


Figure 1: Full Adder

behavior of an exclusive-or gate is:

$$out(X,W,T) \leftarrow xorg(X) \wedge in1(X,U,T) \wedge in2(X,V,T) \wedge norm(X) \wedge xor(U,V,W)$$

The axiom asserts that if X is an exclusive-or gate ($xorg(X)$), the first input of X is U at time T ($in1(X,U,T)$), the second input of X is V at time T ($in2(X,V,T)$), X is normal ($norm(X)$), and the exclusive-or of U and V is W ($xor(U,V,W)$), then the output of X is W at time T ($out(X,W,T)$). In addition we have the facts $xor(0,0,0)$, $xor(0,1,1)$, $xor(1,0,1)$, and $xor(1,1,0)$. The axioms for and gates and or gates are similar.

The following axiom describes the fault mode *stuck-at-0* for all gates:

$$out(X,0,T) \leftarrow in1(X,U,T) \wedge in2(X,V,T) \wedge stuck\text{-}at\text{-}0(X)$$

The axiom for the fault mode *stuck-at-1* is similar. Note that when a gate is assumed to be abnormal, no prediction can be made about its output bit. However, abduction requires that the observations be proved from the component behavioral mode assumptions (including the abnormality assumptions). To overcome this problem, we employ a technique used by Poole to "parameterize" the abnormality assumption as follows [Poole, 1989b]:

$$out(X,W,T) \leftarrow in1(X,U,T) \wedge in2(X,V,T) \wedge ab(X,U,V,W,T)$$

The antecedent $ab(X,U,V,W,T)$ in the rule is to be interpreted as "X is abnormal in such a way that at time T, given input bits U and V, its output bit is W". Note that for any input bits U and V, and any output

bit W, the above axiom always allows us to assume that the component is abnormal by making the assumption $ab(X, U, V, W, T)$. This axiom achieves our objective of being able to prove the output observations from the parameterized abnormality assumption $ab(X, U, V, W, T)$.

So far, the axioms given are not specific to the full adder; they are used to model the behavior of exclusive-or gates, and gates, and or gates. We also need axioms that specify the connections among the gates in the given adder, such as

$$in1(a1, X, T) \leftarrow in1(x1, X, T)$$

as well as facts that identify the five components: $xorg(x1), xorg(x2)$, etc. Furthermore, in order to allow backward-chaining to terminate at the terminal input values of the full adder (these terminal input values cannot be further explained in terms of the other gate values), we need the axiom

$$in1(x1, X, T) \leftarrow given\text{-}in1(x1, X, T)$$

and two other similar axioms for the second input of $x1$ and the first input of $a2$. We let $given\text{-}in1(\ldots)$ (and $given\text{-}in2(\ldots)$) be assumable. They are the auxiliary assumptions, and do not affect the simplicity metric of an explanation.

To assess the performance of ACCEL, we randomly generated 10 scenarios by assuming that the various behavioral modes of each gate occur with the following probabilities: *norm* 65%, *stuck-at*-0 15%, *stuck-at*-1 15%, and *ab* 5%. Each of the 10 scenarios that was actually generated had one or two gates that were faulty, and the scenarios included some where a gate was abnormal (*ab*). For each scenario, we gave ACCEL I/O tuples where the input-output bits of the adder differed from those of a correctly functioning adder. (By an I/O tuple, we mean a particular combination of input and output values of the full adder.) For each I/O tuple, we first gave the three input bits and the two output bits of the adder, and then the output bits of the three gates x1, a1, and a2, in that order. For each scenario, we stopped as soon as the best diagnosis found by ACCEL is the correct diagnosis. We recorded the number of I/O tuples needed to converge on the correct diagnosis for each scenario. On a Sun Sparc 2 workstation, ACCEL took an average of 17 seconds to identify the correct diagnosis for each of the 10 scenarios tested. The average number of I/O tuples needed before the correct diagnosis was found is 2.1.

5.2 DIAGNOSING DYNAMIC SYSTEMS

Much research in model-based diagnosis has focused on diagnosing static, discrete devices like logic circuits. However, many devices and biological systems are continuous and dynamic and require reasoning about changes in behavior over time. Although there has been a great deal of research on modeling and simulating such systems [Kuipers, 1986; Forbus, 1984], there have been few attempts to apply general, model-based diagnostic methods to them. The work of [Ng, 1991; Ng, 1990] attempts to address this deficiency by diagnosing dynamic systems using the consistency-based approach. In this section, we present an abductive approach to diagnosing continuous, dynamic systems.

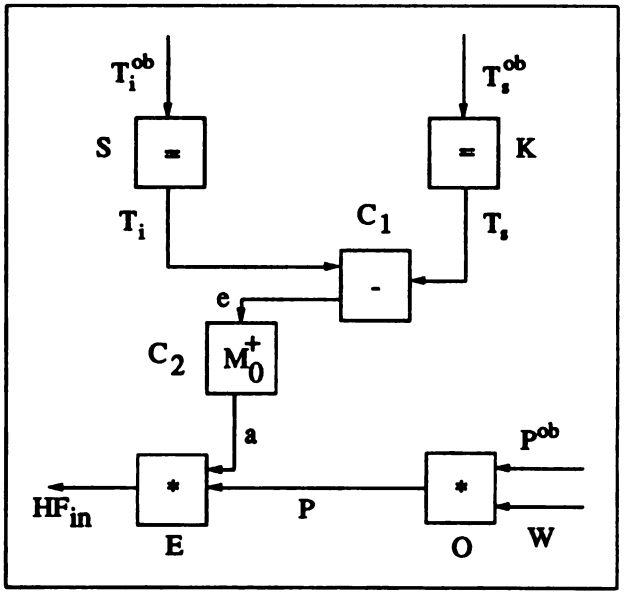


Figure 2: Temperature Controller

We adopt the representation of continuous dynamic systems used in the work of Kuipers' qualitative simulation (QSIM) [Kuipers, 1986]. The continuously changing behavior of a dynamic system over time is represented as a sequence of qualitative states, where a qualitative state consists of the qualitative values of the variables of the system. A qualitative value has two components: a qualitative magnitude (*qmag*) and a qualitative direction (*qdir*). A qualitative magnitude can either be a landmark value or an open interval between two landmark values, where a landmark value is a value of special significance that a variable takes on at some point in time. A qualitative direction can be one of increasing (*inc*), decreasing (*dec*), or steady (*std*).

The behavior of each dynamic system is governed by a set of qualitative constraints. The qualitative constraints on the temperature controller (Figure 2) are as follows (each constraint is preceded by a name identifying that constraint):

1. $S : T_i^{ob} = T_i$;
2. $K : T_s^{ob} = T_s$;
3. $C_1 : T_s - T_i = e$;
4. $C_2 : m_0^+(e) = a$;

5. $O : P^{ob} \cdot W = P$; and
6. $E : a \cdot P = HF_{in}$

The $m_0^+(e) = a$ constraint asserts that there is a strictly monotonically increasing function between e and a. However, the exact form of this monotonic function is unspecified. This accounts for the qualitative nature of the constraint. The purpose of this device is to control the temperature T_i^{ob} in the room, so that if the device is connected to a power source with power P^{ob}, the power switch is turned on (represented as $W = on$), and the temperature T_s^{ob} set by the temperature control knob differs from the temperature T_i^{ob} in the room, heat flow HF_{in} (in the form of hot air or cold air, depending on the direction of temperature difference) will be generated. Furthermore, the amount of heat flow generated is proportional to the temperature difference $T_s^{ob} - T_i^{ob}$.

We have successfully represented QSIM's knowledge about the various qualitative constraints ($=, -, \cdot, /, d/dt, m_0^+$) in Horn-clause axioms in a way suitable for logic-based abductive diagnosis. Since these Horn-clause axioms encode general knowledge about QSIM constraints, they are needed in the diagnosis of every dynamic system. These axioms encode the various qualitative constraints by defining a "holds.constraint-type" predicate for each type of qualitative constraint. For example, one of the 9 axioms that encode the m_0^+ constraint is:

$$holds.m_0^+(F, G, M1, inc, M2, inc)$$
$$\leftarrow$$
$$pos(M1) \wedge pos(M2) \wedge corr\text{-}mag.m_0^+(F, G, M1, M2)$$

The predicate $holds.m_0^+(F, G, M1, D1, M2, D2)$ asserts that $m_0^+(F) = G$ holds with the qualitative value of the variable $F = \langle M1, D1\rangle$ and the qualitative value of the variable $G = \langle M2, D2\rangle$. The predicate $pos(M1)$ ($neg(M1)$) asserts that the qualitative magnitude $M1$ is positive (negative). The predicate $corr\text{-}mag.m_0^+(F, G, M1, M2)$ asserts that $m_0^+(F) = G$ holds with the qualitative magnitude of $F = M1$ and the qualitative magnitude of $G = M2$. In QSIM, $(M1, M2)$ are referred to as corresponding values. The 9 axioms for the m_0^+ constraint cover all the distinct possibilities in which $m_0^+(F) = G$ holds since the qualitative magnitude of F can be positive, negative, or zero, and its qualitative direction can be *inc*, *std*, or *dec*. The other "holds.constraint-type" predicates, $holds.-$, $holds.*$, $holds./$, and $holds.d/dt$, are defined by 39, 97, 70, and 9 axioms, respectively. The axioms for $holds.*(F, G, H, M1, D1, M2, D2, M3, D3)$ ensure that, among other things, the first-order derivative constraint $F \cdot G' + F' \cdot G = H'$ is obeyed. The exact axioms for all the qualitative constraints are listed in [Ng, 1992].

Besides the axioms that encode general QSIM constraints, there are also Horn-clause axioms that encode knowledge about a specific dynamic system. We assume in this paper that a dynamic system malfunctions because of one or more violated constraints, and that the task of mapping from violated constraints to the affected components is done by some other module external to ACCEL. The following axioms describe the normal behavior:

$$qval(ti, M1, D1, T)$$
$$\leftarrow$$
$$norm(s) \wedge qval(ti\text{-}ob, M1, D1, T)$$

$$qval(e, M3, D3, T)$$
$$\leftarrow$$
$$norm(c1) \wedge qval(ts, M1, D1, T) \wedge qval(ti, M2, D2, T) \wedge holds.-(ts, ti, e, M1, D1, M2, D2, M3, D3)$$

The predicate $qval(ti, M1, D1, T)$ asserts that the qualitative value of the variable ti is $\langle M1, D1\rangle$ at time (qualitative state) T. The first axiom asserts that if constraint s is normal, and the qualitative value of $ti\text{-}ob$ is $\langle M1, D1\rangle$ at time T, then the qualitative value of ti is also $\langle M1, D1\rangle$ at time T. This encodes the equality constraint between the variables $ti\text{-}ob$ and ti. The second axiom asserts that if constraint $c1$ is normal, the qualitative value of ts is $\langle M1, D1\rangle$ at time T, the qualitative value of ti is $\langle M2, D2\rangle$ at time T, and $ts - ti = e$ holds with $ts = \langle M1, D1\rangle, ti = \langle M2, D2\rangle, e = \langle M3, D3\rangle$, then the qualitative value of e is $\langle M3, D3\rangle$ at time T. Similar axioms encode the other constraints.

Note that atoms with the predicate *qval* are not assumable. As such, in order to allow backward-chaining to terminate at the terminal input values of a dynamic device (these terminal input values cannot be further explained), we also need the axiom

$$qval(ti\text{-}ob, M1, D1, T) \leftarrow given\text{-}qval(ti\text{-}ob, M1, D1, T)$$

and three other similar axioms for $ts\text{-}ob$, $p\text{-}ob$, and w. We let *given-qval* be assumable. They are part of the "auxiliary" assumptions in an abductive explanation.

Note the directionality in which one qualitative value is explained in terms of other qualitative values. Since abductive diagnosis requires that the input observations (which consists of the qualitative values of the variables of a dynamic system) be *proved*, the axioms are formulated in such a way that the output values (e.g., $qval(hfin, \ldots)$) of a dynamic system can be proved from normality assumptions (e.g., $norm(s)$), fault mode assumptions, and auxiliary assumptions about the input values (e.g., $given\text{-}qval(ti\text{-}ob, \ldots)$) and the qualitative magnitudes and corresponding values of the variables (these are introduced when ACCEL attempts to prove the holds.constraint-type atoms).

We also assume that the components corresponding to the various constraints exhibit the following fault modes: stuck-at-0-std (S, K, C_1, C_2, O, E), stuck-at-roomtemp-std (S), stuck-at-1st-in (C_1, O), and stuck-

at-2nd-in (C_1). Under the fault mode stuck-at-0-std (stuck-at-roomtemp-std), the output of a component is $\langle 0, std \rangle$ ($\langle room\text{-}temp, std \rangle$) regardless of the input values. Under the fault mode stuck-at-1st-in (stuck-at-2nd-in), the output of a component is stuck at its first (second) input. One Horn-clause axiom is used to encode one fault mode, as follows:

$$\begin{aligned} qval(ti, 0, std, T) \leftarrow\ & stuck\text{-}at\text{-}0\text{-}std(s) \wedge \\ & qval(ti\text{-}ob, M1, D1, T) \\ qval(e, M1, D1, T) \leftarrow\ & stuck\text{-}at\text{-}1st\text{-}in(c1) \wedge \\ & qval(ts, M1, D1, T) \wedge \\ & qval(ti, M2, D2, T) \end{aligned}$$

The Horn-clause axioms in ACCEL that represent the qualitative constraints capture the knowledge that QSIM uses to propagate qualitative values across constraints in order to complete the qualitative values of variables in a qualitative state. In ACCEL, such knowledge is used for the purpose of diagnosis. However, since the knowledge is now encoded declaratively, it can also be used for simulation purpose by a forward-chaining inference procedure. In fact, QSIM can be viewed as a special-purpose theorem prover for predicting the behavior of dynamic systems described by qualitative constraints. However, not all of QSIM's knowledge in simulation has been captured in ACCEL. Specifically, knowledge of state transition that QSIM uses to generate the next qualitative state(s) from an initial qualitative state is not encoded in ACCEL, since such knowledge is not needed in diagnosis.

We randomly generated 10 scenarios for the temperature controller where each scenario contains one to two faults and in which no heat flow was generated into the room. For each scenario, we gave the input atoms representing the qualitative values of the variables in the following order: $T_s^{ob}, T_i^{ob}, P^{ob}, W, HF_{in}$ at the initial qualitative state (t_1); $T_s^{ob}, T_i^{ob}, P^{ob}, W, HF_{in}$ at the next distinguished time-point qualitative state (t_2); and the intermediate variables T_s, T_i, e, a, P at state t_2.

In 9 out of the 10 scenarios, ACCEL found the correct diagnosis as its best diagnosis. The one scenario that ACCEL failed to find the best diagnosis has two faults $\{stuck\text{-}at\text{-}0\text{-}std(c1), stuck\text{-}at\text{-}0\text{-}std(c2)\}$. In this case, the best diagnosis that ACCEL found after processing all the intermediate variables is $\{stuck\text{-}at\text{-}0\text{-}std(c1)\}$. This is as it should be, since when $c1$ is stuck at $\langle 0, std \rangle$, the correct behavior of $c2$ if it is normal is to output $a = \langle 0, std \rangle$ at all times, which is indistinguishable from the behavior of $c2$ if it is in the fault mode *stuck-at-0-std*. That $c2$ is in fact faulty would be detected when $c1$ is replaced by a normal, working component and the controller is still found to be malfunctioning. Overall, the average run time per scenario is 4.24 minutes, and the average number of measurements of intermediate variables needed to arrive at the correct diagnosis is 4.4.

We also tested ACCEL on 10 faulty scenarios of the kidney water balance system, a QSIM model of which is given in [Kuipers, 1985; Kuipers, 1991]. The system has 7 qualitative constraints and 10 qualitative variables. Two of the scenarios tested correspond to the disorders Diabetes Insipidus and the Syndrome of Inappropriate Secretion of Anti-Diuretic Hormone (SIADH), which are disorders found in real patients. ACCEL found the correct diagnosis as its best diagnosis in all the 10 scenarios. The average run time per scenario is 6.98 minutes, and the average number of measurements of intermediate variables needed to arrive at the correct diagnosis is 3.7.

6 DISCUSSIONS AND CONCLUSIONS

In this paper, we have presented a general-purpose yet efficient system for making useful abductive inference and solving moderately complex problems in plan recognition and diagnosis. The comprehensive empirical results presented span the tasks of abductive plan recognition, set-covering-based diagnosis, and model-based diagnosis of both discrete and continuous dynamic systems. We believe our approach represents a good trade-off between generality and efficiency — ACCEL is a general-purpose system capable of performing all of the above tasks, yet efficient enough to be of practical utility.

Although ACCEL provides a declarative approach to the generation of explanatory hypotheses, it is often necessary that axioms be formulated carefully so that the system will perform the desired task correctly and efficiently. As in traditional logic programming, it is frequently insufficient to just "state the correct knowledge" and expect the desired answers to be inferred. Appropriate programming methodologies must be developed so that a user knows how to axiomatize a problem to correctly and efficiently compute the desired answers [Poole, 1989a]. This is also true in "abductive logic programming". By successfully applying ACCEL to the tasks of plan recognition and diagnosis, we have demonstrated via many examples *how* a general abductive system can be used to achieve these tasks.

Our empirical results suggest that the best criterion for evaluating explanations varies according to the domain. In diagnosis, the simplicity metric (defined as making the least number of assumptions in an abductive explanation) suffices, whereas in plan recognition for text understanding, because of the need to take into account the importance of text coherence, our coherence metric is a better criterion than simplicity.

Our results in set-covering-based diagnosis show that a general-purpose, logic-based abductive system can effectively represent and efficiently solve large realistic problems previously solved by the set covering method.

Consequently, the desirability of the existing special-purpose approach for such problems is lessened. The logic-based approach is more general and flexible, yet capable of efficiently solving problems in this more restrictive class.

Our empirical results in model-based diagnosis indicate that ACCEL's abductive approach can diagnose devices and systems previously solved by consistency-based methods. Although the work of [Poole, 1988; Poole, 1989b; Konolige, 1992] has revealed some interesting relationships between consistency-based and abductive diagnosis, the extent to which the two approaches coincide and differ, especially in practical terms such as ease of representation and diagnostic efficiency, requires further investigation. In addition, our research does not focus on intelligently gathering additional measurements to further differentiate and narrow the diagnostic candidates. Future work needs to extend ACCEL to incorporate intelligent experimentation.

In summary, this paper has demonstrated via an implemented system that general and efficient abduction for the tasks of plan recognition and diagnosis is indeed possible, and the future holds much promise for such a general abductive approach to explanation.

Acknowledgements

The first author was supported by an IBM Graduate Fellowship. This research was also supported by the NASA Ames Research Center under grant NCC 2-629. We are indebted to Dr. Stanley Tuhrim of Mount Sinai School of Medicine, New York, and Dr. James Reggia of the University of Maryland at College Park who kindly provided us the knowledge base on brain damage and the 50 patient test cases.

References

[Allen, 1987] James F. Allen. *Natural Language Understanding.* Benjamin/Cummings, Menlo Park, CA, 1987.

[Charniak and McDermott, 1985] Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence.* Addison Wesley, Reading, MA, 1985.

[Davis, 1984] Randall Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24:347–410, 1984.

[de Kleer and Williams, 1987] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[de Kleer and Williams, 1989] Johan de Kleer and Brian C. Williams. Diagnosis with behavioral modes. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1324–1330, Detroit, MI, 1989.

[Dvorak and Kuipers, 1989] Daniel Dvorak and Benjamin J. Kuipers. Model-based monitoring of dynamic systems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1238–1243, Detroit, MI, 1989.

[Forbus, 1984] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[Goldman, 1990] Robert P. Goldman. *A Probabilistic Approach to Language Understanding.* PhD thesis, Department of Computer Science, Brown University, Providence, RI, December 1990. Technical Report CS-90-34.

[Grice, 1975] H. P. Grice. Logic and conversation. In P. Cole and J. Morgan, editors, *Syntax and Semantics 3: Speech Acts*, pages 41–58. Academic Press, New York, 1975.

[Konolige, 1992] Kurt Konolige. Abduction versus closure in causal theories. *Artificial Intelligence*, 53:255–272, 1992.

[Kuipers, 1985] Benjamin J. Kuipers. Qualitative simulation in medical physiology: A progress report. Technical Report MIT/LCS/TM-280, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, June 1985.

[Kuipers, 1986] Benjamin J. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.

[Kuipers, 1991] Benjamin J. Kuipers. Qualitative reasoning: Modeling and simulation with incomplete knowledge. Book draft, May 1991.

[Lehnert and Sundheim, 1991] Wendy Lehnert and Beth Sundheim. A performance evaluation of text-analysis technologies. *AI Magazine*, 12(3):81–94, 1991.

[Levesque, 1989] Hector J. Levesque. A knowledge-level account of abduction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1061–1067, Detroit, MI, 1989.

[Ng and Mooney, 1990] Hwee Tou Ng and Raymond J. Mooney. On the role of coherence in abductive explanation. In *Proceedings of the National Conference on Artificial Intelligence*, pages 337–342, Boston, MA, 1990.

[Ng and Mooney, 1991] Hwee Tou Ng and Raymond J. Mooney. An efficient first-order Horn-clause abduction system based on the ATMS. In *Proceedings of the National Conference on Artificial Intelligence*, pages 494–499, Anaheim, CA, 1991.

[Ng, 1990] Hwee Tou Ng. Model-based, multiple fault diagnosis of time-varying, continuous physical devices. In *Proceedings of the Sixth IEEE Conference on Artificial Intelligence Applications*, pages 9–15, Santa Barbara, CA, 1990. To appear in *Readings in Model-based Diagnosis*, edited by Walter Hamscher, Luca Console, and Johan de Kleer.

[Ng, 1991] Hwee Tou Ng. Model-based, multiple-fault diagnosis of dynamic, continuous physical devices. *IEEE Expert*, 6(6):38–43, 1991.

[Ng, 1992] Hwee Tou Ng. *A General Abductive System with Application to Plan Recognition and Diagnosis.* PhD thesis, Department of Computer Sciences, University of Texas at Austin, Austin, TX, May 1992.

[Norvig and Wilensky, 1990] Peter Norvig and Robert Wilensky. A critical evaluation of commensurable abduction models for semantic interpretation. In *Proceedings of the Thirteenth International Conference on Computational Linguistics*, Helsinki, Finland, August 1990.

[Peng and Reggia, 1990] Yun Peng and James A. Reggia. *Abductive Inference Models for Diagnostic Problem-Solving.* Springer-Verlag, New York, 1990.

[Poole, 1988] David Poole. Representing knowledge for logic-based diagnosis. In *Proceedings of the International Conference on Fifth Generation Computing Systems*, pages 1282–1290, Tokyo, Japan, 1988.

[Poole, 1989a] David Poole. A methodology for using a default and abductive reasoning system. Technical Report 89-20, Department of Computer Science, University of British Columbia, Vancouver, B.C., Canada, September 1989.

[Poole, 1989b] David Poole. Normality and faults in logic-based diagnosis. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1304–1310, Detroit, MI, 1989.

[Reggia *et al.*, 1985] James A. Reggia, Dana S. Nau, and Pearl Y. Wang. A formal model of diagnostic inference. I. problem formulation and decomposition. *Information Sciences*, 37:227–256, 1985.

[Reiter, 1987] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[Selman and Levesque, 1990] Bart Selman and Hector J. Levesque. Abductive and default reasoning: A computational core. In *Proceedings of the National Conference on Artificial Intelligence*, pages 343–348, Boston, MA, 1990.

[Stickel, 1988] Mark E. Stickel. A Prolog-like inference system for computing minimum-cost abductive explanations in natural-language interpretation. Technical Note 451, SRI International, September 1988.

[Struss and Dressler, 1989] Peter Struss and Oskar Dressler. Physical negation — integrating fault models into the general diagnostic engine. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1318–1323, Detroit, MI, 1989.

[Tuhrim *et al.*, 1991] Stanley Tuhrim, James Reggia, and Sharon Goodall. An experimental study of criteria for hypothesis plausibility. *Journal of Experimental and Theoretical Artificial Intelligence*, 3:129–144, 1991.

Using Default and Causal Reasoning in Diagnosis

Kurt Konolige
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
konolige@ai.sri.com

Abstract

We present a theory of default reasoning that is specifically targeted to causal domains. These domains encompass a wide variety of current applications of default reasoning, but here we concentrate on model-based diagnosis. The theory is unique in that it integrates a formal notion of causality with nonmonotonic reasoning techniques based on default logic and abduction. The main structure of the theory is a default causal net (DCN) representing the causal connections among propositions in the domain. The causal net provides a framework for the two nonmonotonic reasoning techniques of assuming defaults and generating explanations for observations, allowing them to be combined in a principled way.

1 Introduction

Knowledge of causation is an important part of commonsense reasoning. We use cause-and-effect analysis to understand everything from why we caught the flu to how to make a video recorder save our favorite TV show. Our facility can be characterized by a combination of two main capabilities: the ability to predict the outcome of a set of causative events; and the ability to explain given facts by postulating a set of determining causes.

Causal reasoning is most effective when combined with another type of commonsense reasoning, the assumption of normal conditions under which the reasoning is valid. While the existence of these assumptions has been recognized in the philosophical literature on causation, relatively little attention has been paid to the practical aspects of reasoning involving causation and normal conditions: for example, what happens when some causes affect the conditions on which other causes depend?

In this paper we present a theory that integrates causal and default reasoning. The main structure of the theory is a default causal net (DCN) representing the causal connections among propositions in the domain. Default causal nets offer significant representational advantages over current formal model-based diagnosis theories.

There have been other proposals to use causation in formal theories; perhaps the closest to our approach is the use of causation in Bayes nets [Pearl, 1988]. Our goal is similar: to use causation as a structuring concept to guide the application of a formalism that is, in effect, too general. Large probability distributions are difficult to define in particular domains, and causation provides an abstract view that structures the probability space with independence assumptions. Analogously, we use causation to structure logical theories of defaults, furnishing both a guide to the application of default reasoning, and a formal system that functions at a useful level of abstraction. In addition, we find that a concept of causation provides other benefits: a richer notion of explanation, and a computational regime. More specifically, we claim the following advantages over current model-based theories of diagnosis [de Kleer *et al.*, 1990, Reiter, 1987].

- Our theory distinguishes between the strong explanation of the cause of an observation versus the weaker explanation of an excuse for the consistency of the observation.
- Partial fault models [de Kleer and Williams, 1989] are allowed; information about fault modes can lead to stronger explanations, but complete information is not required.
- Preferences among explanations based on causal relations in DCNs can yield better diagnoses than current model-based theories.
- Because it is based on abductive reasoning, DCNs admit causal influences that are neither normal or abnormal, but neutral.

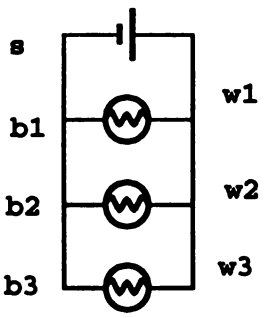


Figure 1: Three Bulbs Example

To show the utility of causal reasoning in model-based diagnosis, consider the example of Figure 1 (adapted from [Struss and Dressler, 1989]). There are three bulbs in parallel with a source. Let us suppose for simplicity that the wires always behave correctly. If we observe that b_3 is on while b_1 and b_2 are off, we conclude that the battery and b_3 are working correctly, while b_1 and b_2 are broken. But using a straightforward encoding of the correct behavior of the system, and applying Reiter's model-based theory of diagnosis [Reiter, 1987], we get two diagnoses: the correct one, and one that says both s and b_3 are abnormal. The reason is that an abnormal source is a reason for b_1 and b_2 being off, and an "abnormal" b_3 is on even without any voltage being present! This was noted by Struss and Dressler [Struss and Dressler, 1989], and they proposed adding fault models (an encoding of what happens when a component is abnormal) to the system description; another, slightly simpler solution is to add physical impossibility axioms [Friedrich *et al.*, 1990]. Both solutions have problems. We may not know all the fault modes; and in such cases fault models are useless for model-based diagnosis. And for complex systems, it may be difficult to determine the physical impossibility axioms, since we must enumerate all combinations of atoms that are physically impossible.

A causal theory offers a more reasoned approach to diagnosis, simulating the intuitive process by which we arrive at the correct solution. For the bulb example, we represent that the source normally causes b_1, b_2, and b_3 to be on. In seeking to explain the observations, the most normal explanation is that the source and b_3 are working correctly (thereby explaining causally why b_3 is on), and that b_1 and b_2 are broken, thereby excusing the normal causation of b_1 and b_2 being on. There is no need for fault models or physical impossibility axioms: the process of finding causal explanations and excuses generates the correct diagnosis.

In the next section we present the main components of our theory of Default Causal Nets (DCNs): causes, normal conditions, definitions, and correlations. We show how DCNs can be used as a representation of causal knowledge, and how explanations, excuses and predictions can be generated from them. We have implemented the theory using a modification of the techniques available with an ATMS [de Kleer, 1986].

2 Default causal nets

Default causal nets (DCNs) are a formal structure that encodes the concepts of causation, correlation, and defaults. They consist of a causal theory R, a definitional theory D, and a correlation or integrity theory I. In addition there are distinguished sets of propositions C (the primitive causes) and N (the normal conditions). The term "net" is used in analogy with Bayesian nets, because the main structuring concept is the causal relation embodied in R.

Definition 1 (Default Causal Net)
A default causal net is a tuple $(\mathrm{R}, \mathrm{D}, \mathrm{I}, C, N)$, *where* R *is a Horn theory,* D *and* I *are first-order theories, and* C *and* N *are disjoint sets of atoms.*

2.1 Causation

Formally, we understand causation to be a primitive relation among propositions. By "primitive" we mean that, as far as DCNs are concerned, the causation relation is part of the parameterization of the net, and is not derived from any other concepts. This is unlike the approach of Shoham [Shoham, 1987], for example, which is a formal theory of causation in terms of other concepts. Our approach leaves unanswered questions about how to identify causation in a given domain, the relation of causation to time, and various other difficulties about the nature and properties of causation. We will have very little more to say here about these important questions, other than giving the most basic (and hopefully noncontroversial) properties (but see Section 4.2). These properties suffice to develop the basic outlines of DCNs; further investigations will have to confront some of the harder problems.

To represent the causal relation, we use a propositional definite clause theory R over a first-order language $\mathcal{L}$. This theory consists of a set of implications

$$a_1 \cdots a_n \supset b \,.$$

where each of a_i and b is a ground atom of $\mathcal{L}$. If A is a set of propositions, then we say that an atom b is caused by A if there is a proof of b from A in R; we write this as $A \vdash_{\mathrm{R}} b$. A is a minimal cause for c if there is no other cause A' for c such that $A' \subset A$.

Example 1 *A variation of the 3-bulb example is diagrammed in Figure 2. There is a switch that can be either open or closed. For each of the other components* c_i, *the proposition* $ok(c_i)$ *means that the component is*

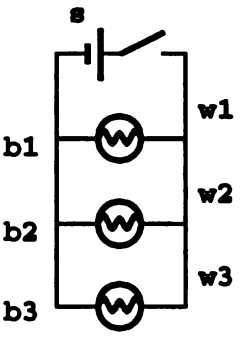


Figure 2: Three Bulbs with a Switch

working, and $ab(c_i)$ *that it is broken. The theory* R *is:*

$$\begin{aligned} &closed, ok(s), ok(w_1), ok(b_1) \supset on(b_1) \\ &closed, ok(s), ok(w_1), ok(w_2), ok(b_2) \supset on(b_2) \\ &closed, ok(s), ok(w_1), ok(w_2), ok(w_3), ok(b_3) \supset on(b_3) \end{aligned} \quad (1)$$

We have not listed any fault models, although we could. Here is a partial fault model that we will use in some examples.

$$\begin{array}{ll} open \supset off(b_1) & open \supset off(b_2) \\ ab(b_1) \supset off(b_1) & ab(b_2) \supset off(b_2) \\ ab(w_1) \supset off(b_1) & ab(w_2) \supset off(b_2) \end{array} \quad (2)$$

The partial fault model is also part of the relation R, *since it represents causation in the abnormal functioning of the device.*

The primitive causes C *are* $\{open, closed, ab(c_i)\}$. *Note that, unlike model-based diagnosis, there can be causes other than the normal or abnormal functioning of a component. This is useful in representing neutral situations, e.g., the switch is not normally either closed or open, but can be hypothesized as either in order to explain the observations. The propositions* $ok(x)$ *are not listed as primitive causes; they are normal conditions, explained below.*

The important part of the causal relation is that it captures the functional dependence of the domain variables. If we want to turn b_1 on, then we can close the switch and make sure that s and w_1 are working correctly. On the other hand, we cannot make b_1 be on as a means of causing the switch to close. Of course, if we observe b_1 to be on, then we can infer that the switch is closed; but it is not possible to *plan* to change the position of the switch by the primitive action of making the bulb be on. This illustrates the difference between a causal relation and a merely correlational one. The causal relation is asymmetric: given that c causes d, it is not necessarily the case that $\neg d$ causes $\neg c$; while if d is positively correlated with c, $\neg c$ is positively correlated with $\neg d$. Deduction in a definite clause theory is one way to represent an asymmetric causal relation.

2.2 Definitions and correlations

Besides causation, there are other types of relations connecting propositions. Definitional information relates propositions that have defined relations within a domain, e.g., "a 40-watt bulb is a type of bulb" or "abnormal is the opposite of normal." Definitions can obviously interact with causation, since from "a broken 40-watt bulb caused the problem" we can infer "a broken bulb caused the problem." For our purposes, we limit definitions to information about complementary propositions. Definitional relations are represented by a first-order theory D; for the bulbs example of Figure 2, it contains the propositions:

$$\begin{aligned} open &\equiv \neg closed \\ ab(c_i) &\equiv \neg ok(c_i) \\ on(b_i) &\equiv \neg off(b_i) \end{aligned} \quad (3)$$

In this paper, we will just use the definitional theory to give the complements of propositions. If $p \vdash_D \neg q$, then we say that q is the complement of p, and write it as $\overline{p}$.

Information about co-occurrences is another form of non-causal information in a domain, e.g., "Whenever I clean my car it rains." Correlations can be used to make predictions, but do not contribute to causal explanations. Correlations are represented by a first-order theory I (for *integrity* theory). All causation and definition relations are also correlational. We enforce this restriction by demanding that R $\subseteq$ I and D $\subseteq$ I.

Example 2 *Continuing the bulbs example, suppose we know that whenever* b_1 *is off and is not broken, the other bulbs must be off too. We represent this as*

$$off(b_1) \wedge ok(b_1) \supset off(b_2) \wedge off(b_3) \in \mathrm{I} \quad (4)$$

Correlations may come from many different sources. As in the case of this example, there may be underlying but unknown causes that link together several propositions. Or we may have experiential knowledge that is the converse of causation: whenever the road is wet, it normally rained the previous night.

A proposition q is correlationally inferred from a set of propositions A if it follows logically from the correlational theory and A; we write $A \vdash_I q$. For example, $off(b_2)$ is inferred from $A = \{ok(b_1), off(b_1)\}$, but it is not caused by A. If A causes q, then it also infers q, since R $\subseteq$ I. Note that, unlike the case with the causal relation, the material conditional can be used in for "backwards" inference, e.g., if $on(b_2)$ is true, we can infer that one of $ab(b_1)$ or $on(b_1)$ is true.

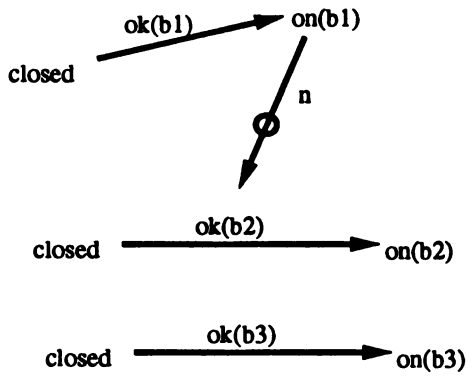


Figure 3: Causal Directionality

2.3 Normal conditions

Normal conditions are propositions that are normally assumed to hold. They generally represent either the normal functioning of a component, or a complex set of conditions, e.g., "if the key is turned and *everything is normal*, the car will start." Formally, normal conditions are a set of ground atoms N that are not primitive causes. Primitive causes are hypotheses that incur a cost to assume; normal conditions are "free" and assumed to hold by default.

Example 3 *Continuing the bulbs example, we let the set of normal conditions* $N = \{ok(c_i)\}$. *In this case, the normal conditions just describe the correct functioning of the components. We can define other types of normal conditions, for example to relate causation among abnormal components. Suppose that normally when* b_1 *is on, it causes* b_2 *to fail. We would write:*

$$n \wedge on(b_1) \supset ab(b_2) \qquad (5)$$

as part of the causal theory R, *where* $n \in N$ *is a new proposition reflecting a normal causal relation between* b_1 *and* b_2. *As we will show later, such causal relations can be used to specify priorities among explanations.*

Identifying normal conditions is the key to default reasoning in causal theories. We will seek to explain a set of observations by hypothesizing causes that are as "normal" as possible, that is, conflict with the fewest normal conditions.

It is helpful to view the causal relation and normal conditions as a directed graph. For example, the normal functioning of the bulbs (Equation 1) and the failure mode just given (Equation 5) can be diagrammed as in Figure 3. The arrows show the causal connections among propositions, annotated with their normal conditions (for simplicity we have omitted some irrelevant normal conditions). The circled arrow indicates that bulb 1 being *on* is the cause of an abnormal condition with bulb 2. The causal directionality is clear from the diagram.

The choice of what conditions are assumed to be "normal" or part of the causal background is an important part of the information provided by the theory developer. Depending on the task and the level of expertise of the developer, very different choices could be made, even in the same domain. For example, a typical driver might infer that turning the key causes the car to start, given the normal condition that the car is ok. A car mechanic might have a more detailed causal view: turning the key and having a charged battery causes the car to start, assuming the starter motor is working correctly.

2.4 Explanations

We now have all of the elements necessary to develop the inference operation of explanation within DCNs.

Definition 2 (Explanation)
An explanation for an observation set O *is a set of causes and normal conditions* $A \subseteq C \cup N$ *such that* $A \vdash_R O$ *and* $A \cup O \not\vdash_I \bot$.

Example 4 *To illustrate the concept of explanation, we consider the bulbs theory containing the normal causal rules (1) together with the fault model (2). The fault model is necessary to provide interesting explanations of nonnormal behavior. Suppose we make the observation that bulb* b_1 *is not lit:* $off(b_1)$. *There are several explanations for this proposition.*

$$open, ok(s), ok(b_1), ok(w_1)$$
$$closed, ok(s), ab(b_1)$$
etc.

There are usually many explanations for a given observation set, and we seek intuitively preferred explanations. To find these, we filter all explanations by a two-step process.

1. Normal explanation: those explanations that satisfy a maximum set of normal conditions.
2. Ideal explanation: normal explanations that have a minimal number of primitive causes.

The concept of a normal explanation is complicated by the presence of causation. An abnormal condition may be caused by the explanation; when this happens, we say that the normal condition is *exempted*. A normal explanation should either consistently include or exempt as many normal conditions as possible. Here we are using the concept of causation to structure the defaults. If a normal condition is not contained in an explanation, it counts against the explanation, *unless* the corresponding abnormal condition is exempted.

Definition 3 (Adjunct)
Let A be an explanation for observation set O. The adjunct *of A is a set of normal conditions defined as follows.*

- *If the complement $\overline{x}$ of a normal condition x is in A, then x is in the adjunct.*
- *If a normal condition x is not in A, and $A \not\vdash_R \overline{x}$, then x is in the adjunct.*

A normal explanation *for O is one whose adjunct is not contained in the adjunct of any other explanation for O. An ideal explanation is a normal one that is subset-minimal in the primitive causes.*

Example 5 *As in the previous example, consider the bulbs theory (1) together with the fault model (2). Again, if we make the observation that bulb b_1 is off, we have several candidates for normal explanations:*

Explanation	*Adjunct*
$ok(s), open, ok(w_1), ok(b_1) \cdots$	*none*
$ok(s), ab(w_1), ok(b_1) \cdots$	$ok(w_1)$
$ok(s), ok(w_1), ab(b_1) \cdots$	$ok(b_1)$

Of these, the minimal adjunct is the first, and so it is the normal and ideal explanation of $off(b_1)$: the switch is open, and all components are normal.

This example illustrates one property of normal explanations: as many normal conditions are assumed to hold as possible. The switch can be either open or closed; if we assume that it is open, then we have an explanation for b_1 being off that is consistent with the normal functioning of the circuit. Any other explanation will force us to assume that some component is functioning abnormally. So, normal explanations consist of a set of primitive causes that explain the observations, and at the same time respect our ideas about what normally occurs as much as possible.

In this example, there were no interesting causal relations between normal conditions. In the definition of adjunct, we used the principle of *causal exemption*: if an abnormal condition is caused by the hypothesized explanation, then it is exempted from consideration in finding the "most normal" explanation. The following example illustrates this point.

Example 6 *Consider the same fault model as in Example 5 with an initial condition closed and the additional causal rule (5): $n \wedge on(b_1) \supset ab(b_2)$. There are several candidates for normal explanations of $\{off(b_2)\}$:*

Explanation	*Adjunct*
$n, ok(s), ok(w_1), ok(b_1), ok(w_2) \cdots$	*none*
$n, ok(s), ok(w_1), ok(b_1), ab(w_2) \cdots$	$ok(w_2)$
$ok(s), ok(w_1), ok(b_1), ok(w_2), ab(b_2) \cdots$	$ok(b_2), n$
etc.	

Of these, the first is the only normal explanation, and hence ideal. The reason it has an empty adjunct is that the normal conditions and closed cause $on(b_1)$, which in turn causes $ab(b_2)$, exempting the normal condition $ok(b_2)$. Every other explanation violates at least one normal condition without exempting it. This makes intuitive sense: if the switch is closed, we expect b_1 to be on, causing b_2 to be broken and off.

This example illustrates how directionality in the causal relation is important in producing causal exemption. Referring back to Figure (3), it is easy to see from following the causal arrows that *closed*, $ok(b_1)$ and n are a cause of $ab(b_2)$. On the other hand, *closed* and $ok(b_2)$ are inconsistent with n and $ok(b_1)$, but they do not cause the complement of either of these normal conditions.

2.5 Excuses

One problem with causal explanations is that they always require a causal model that infers the observations. Without the partial fault model of Equation (2), for example, there are no explanations for why the bulbs are off when the switch is closed. In many cases, it may not be possible to find a causal explanation for all the observations, given a causal theory with incomplete fault models. In this situation the weaker concept of an excuse might be appropriate. The idea here is that the system, if it were functioning normally, would produce output inconsistent with the observations. By hypothesizing some primitive causes, the state of the system can be changed so that it no longer conflicts with the observations.

Definition 4 (Excuse)
An excuse *for observations O is a set of causes and normal conditions $A \subseteq C \cup N$ such that $A \cup O \not\vdash_I \bot$.*

Excuses are like explanations, except there is no necessary causal relation to the observations.

Fact 1 *Every explanation of O is an excuse for O, but not necessarily the converse.*

Normal and ideal excuses can be defined in exactly the same manner as for explanations. Note that, although we do not use the causation relation to infer the observations, we still use it to give preferences on the normal conditions present in excuses, in exactly the same manner as for explanations.

With excuses, we do not need to define fault models in order to "excuse" a set of observations. Excuses are useful precisely in those cases where we do not have enough information to make a predictive fault model; all we know is that some component is faulty, and we no longer can predict that the behavior of the system is at odds with the observations.

Example 7 *Consider the simple bulbs theory from Equation (1), with no fault models. There is no explanation for* $off(b_1)$*, but there are several excuses:*

Excuse	*Adjunct*
$ok(s), ok(w_1), ok(b_1) \cdots$	*none*
$open, ok(s), ok(w_1), ok(b_1) \cdots$	*none*
$closed, ok(w_1), ok(b_1) \cdots$	$ok(s)$
$closed, ok(s), ok(b_1) \cdots$	$ok(w_1)$
etc.	

The first two of these are normal excuses because they have minimal adjuncts. Of these, the first is ideal, because it does not have any assumed primitive causes.

Excuses are essentially the idea behind Reiter's model-based diagnosis method [Reiter, 1987]. In fact, we can show that his concept of diagnosis is exactly the concept of an excuse with no causal knowledge. In Reiter's theory, a system is a tuple (SD, CMPS), where SD is a first-order theory describing the system and CMPS is a set of component names.

Definition 5 *A* diagnosis *for observations* O *relative to a system* (SD, CMPS) *is a minimal set of components* $\Delta \subseteq$ CMPS *such that*

$$\text{SD} \cup O \cup \{ab(c) \mid c \in \Delta\} \cup \{\neg ab(c) \mid c \in \text{CMPS} - \Delta\}$$

is consistent.

In DCN terms, the system description corresponds to the correlational theory, the abnormalities are primitive causes, and their complements are normal conditions. The causal relation is empty; according to our analysis, Reiter's theory does not distinguish causation from correlation, all relationships are treated as correlations. In this case there are simplifications in the DCN: the adjunct of an excuse A is just the set of normal conditions not in A, and a normal excuse contains a maximally consistent set of normal conditions. Normal excuses are exactly Reiter's diagnoses.

Fact 2 (Diagnosis theory)
Let (SD, CMPS) *be a system. Construct a corresponding DCN as follows:*

$$\begin{aligned} \text{R} &= \emptyset \\ \text{I} &= \text{SD} \\ \text{D} &= \{ab(c) \equiv \neg ok(c) \mid c \in \text{CMPS}\} \\ C &= \{ab(c) \mid c \in \text{CMPS}\} \\ N &= \{ok(c) \mid c \in \text{CMPS}\} \end{aligned}$$

Then Δ *is a diagnosis of* O *with respect to* (SD, CMPS) *if and only if* $\{ok(c) \mid c \in \text{CMPS} - \Delta\}$ *is a normal excuse for* O *in the corresponding DCN.*

Because the model-based theory does not have a causation relation, causal exemption is not possible. Suppose we reconsider Example 6, in which the switch is assumed closed and the observation is $off(b_2)$. Now assume that the causal rules (1) and (5) are purely correlational, and the causal theory is empty. Then we have the following excuses for $off(b_2)$:

Excuse	Adjunct
$n, ok(s), ok(w_1), ok(b_1), ok(w_2) \cdots$	$ok(b_2)$
$n, ok(s), ok(w_1), ab(b_1), ok(w_2), ok(b_2) \cdots$	$ok(b_1)$
$ok(s), ok(w_1), ok(b_1), ok(w_2), ok(b_2) \cdots$	n
$ok(s), ab(w_1), ok(b_1), ok(w_2), ab(b_2) \cdots$	$ok(b_2), ok(w_1), n$

etc.

The first three of these are normal excuses. Either bulb b_2 is broken, or something is wrong with bulb b_1, or the normal connection between b_1 and b_2 doesn't hold. There are also other normal excuses, in which various wires or the source is assumed to be malfunctioning. The correct causal solution is still there, but cannot be distinguished from the other excuses.

2.6 Lenient explanations

Excuses are weaker than explanations, and we seek explanations whenever possible, as being more informative. While there may be no explanations for every member of an observation set O, it may be possible to find an explanation for a subset of O, while excusing the rest.

Definition 6 (Lenient explanation)
Let $O' \subseteq O$ *be a maximal subset of* O *for which an explanation exists. Then a* lenient explanation *for* O *is a set of causes and normal conditions* $A \subseteq C \cup N$ *such that* A *is an explanation for* O' *and an excuse for* $O - O'$.

Obviously, if O has a causal explanation, all lenient explanations are explanations. As with ordinary explanations and excuses, we can define the concept of a normal lenient explanation and ideal lenient explanation.

Example 8 *This is the original bulb example cited in the Introduction; its causal relation is given by Equation (1), with the proposition closed removed since there is no switch.*

Suppose we observe that bulbs one and two are off, and bulb three is on ($O = \{off(b_1), off(b_2), on(b_3)\}$). *There is no explanation for* $off(b_1)$ *and* $off(b_2)$*, but there is for* $on(b_3)$. *So any lenient explanation of* O *will include* $\{ok(s), ok(w_1), ok(w_2), ok(w_3), ok(b_3)\}$. *In fact this is the only explanation, since adding either* $ok(b_1)$ *or* $ok(b_2)$ *will contradict the observations. It is lenient, normal, and ideal.*

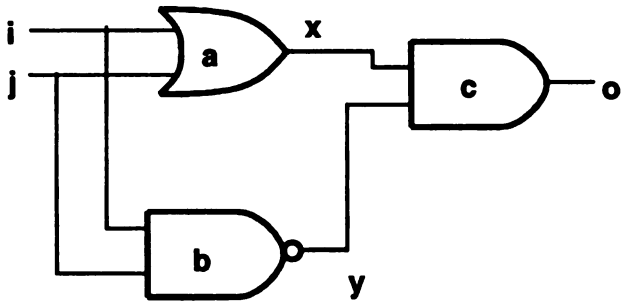


Figure 4: An XOR Circuit

Looking at the excuses for O, we get the following normal ones:

1. $ok(w_1), ok(w_2), ok(w_3), ok(b_1), ok(b_2), ok(b_3)$
2. $ok(s), ok(w_2), ok(w_3), ok(b_1), ok(b_2), ok(b_3)$
3. $ok(s), ok(w_1), ok(w_3), ok(b_2), ok(b_3)$
4. $ok(s), ok(w_1), ok(w_2), ok(w_3), ok(b_3)$

Each of these corresponds to a maximal set of normal conditions that does not infer $on(b_1)$ or $on(b_2)$ or $off(b_3)$; but only the last one corresponds to the correct causal explanation.

The lenient ideal explanations (LIEs) of an observation set are the ones we usually want. LIEs are generated by the following steps:

1. Find the lenient explanations of O.
2. Of these, choose the ones that are have a minimal adjunct.
3. Of these, choose the ones that have minimal non-normal causes. These are the lenient ideal explanations of O.

3 Computational methods

We develop some computational methods that can be applied to generate LIE's for an observation set. These methods are similar to the minimal conflict methods of diagnostic theories [de Kleer *et al.*, 1990, Reiter, 1987]. We use the xor function circuit diagrammed in Figure 4 as an example. There are three gates, a, b, and c, with two inputs (i, j) and one output (o). We let the atoms i, j, o, x and y stand for circuit logic levels, so that i means input i is one, $\bar{i}$ that it is zero. We have the following causal relation R:

$i, ok(a) \supset x$	$x, y, ok(c) \supset o$	$i, j, ok(b) \supset \bar{y}$
$j, ok(a) \supset x$	$\bar{x}, ok(c) \supset \bar{o}$	$\bar{i}, ok(b) \supset y$
$\bar{i}, \bar{j}, ok(a) \supset \bar{x}$	$\bar{y}, ok(c) \supset \bar{o}$	$\bar{j}, ok(b) \supset y$
$ab(a) \supset \bar{x}$	$ab(c) \supset \bar{o}$	$ab(b) \supset \bar{y}$

The normal conditions are $N = \{ok(a), ok(b), ok(c)\}$. Whenever a gate fails, its output is always stuck at zero. The primitive causes are $C = \{i, \bar{i}, j, \bar{j}, ab(a), ab(b), ab(c)\}$.

3.1 Minimal conflicts and regular explanations

The first step is to consider compact ways to represent normal causal explanations. One idea is to just consider explanations that are minimal in causes, which is a large reduction in the search space. Normal explanations may not be minimal in primitive causes; nevertheless under certain circumstances we can represent all normal explanations as a combination of a minimal explanation and a maximally consistent set of normal conditions.

We will use only a single atom g as the observation; an observation set O can be accommodated by making a new atom q, adding the causal rule

$$o_1 \wedge \cdots \wedge o_n \supset q\ ,$$

and taking q as the single observation.

Recall that the adjunct of a explanation A for g is the set of normal conditions not contained in or exempted by A. The exemptions are a complicating factor, since they may introduce causes into A that have nothing to do with the derivation of g. Let us call an explanation A asserting no exemptions a *regular* explanation. The adjunct of a regular explanation A has every normal condition not contained in A. Also, let us define an *extension* of an explanation A as A together with maximally consistent set of normal conditions. A *minimal explanation* A is one that has a subset-minimal set of primitive causes. The following decomposition relation holds.

Fact 3 *Every regular explanation A can be decomposed into a minimal explanation A' and a set of normal conditions W, such that $A = A' \cup W$. If A is normal, then it is an extension of A'.*

This result is encouraging. It suggests that we can find normal (regular) explanations for g by looking at its minimal explanations and comparing their extensions.

Fact 4 *If all the normal explanations of g are regular, then they are exactly the extensions of the minimal explanations of g.*

Example *We consider the xor circuit with the additional correlation that if c fails, so does one of a or b: the correlational theory contains $ab(c) \supset (ab(a) \vee ab(b))$. Assuming $\bar{j}$ and i as initial conditions, there are three minimal explanations of $\bar{o}$. We list these with*

the normal conditions of their extensions.

	Minimal expl	Extensions
1.	$\bar{j}, i, \mathbf{ab}(c)$	$ok(a)$
		$ok(b)$
2.	$\bar{j}, i, \mathbf{ab}(a), ok(c)$	$ok(c), ok(b)$
3.	$\bar{j}, i, \mathbf{ab}(b), ok(c)$	$ok(c), ok(a)$

The normal explanations are the extensions of 2 and 3.

Rather than computing all normal extensions of the minimal explanations of g, we can just compute the minimal conflicts and candidates [de Kleer and Williams, 1987, Reiter, 1987], and use these to generate the normal explanations directly. If there are many normal conditions, the minimal conflict encoding is usually a much more efficient means of encoding all of the extensions of a minimal explanation.

The definitions follow closely those of [de Kleer and Williams, 1987, Reiter, 1987], but are relativized to a given causal explanation. A *minimal conflict* for an explanation A is a minimal set of normal conditions that is inconsistent with $A \cup I$. If A' is an extension of A, the set of normal conditions not in A' is called a *candidate* of A. The candidates of A can be readily generated from the minimal conflicts of A (candidates are called *hitting sets* in [Reiter, 1987]). To form the candidates, we pick one element from each minimal conflict.

Fact 5 *Suppose all the normal explanations of g are regular. Let Σ be the minimal explanations of g. The normal explanations of g are exactly the extensions of elements of Σ whose candidates are subset-minimal over Σ.*

From this result, we need only compare the candidates of the minimal explanations for g, in order to find the normal ones.

Example 10 *Considering again the xor circuit with the added correlation of example 9, and assuming $\bar{j}$ and i as initial conditions, we list the minimal conflicts and candidates for each of the three minimal explanations of $\bar{o}$.*

	Minimal expl	Conflicts	Candidates
1.	$\bar{j}, i, \mathbf{ab}(c)$	$ok(a), ok(b)$	$ok(c), ok(a)$
		$ok(c)$	$ok(c), ok(b)$
2.	$\bar{j}, i, \mathbf{ab}(a), ok(c)$	$ok(a)$	$ok(a)$
3.	$\bar{j}, i, \mathbf{ab}(b), ok(c)$	$ok(b)$	$ok(b)$

Explanations 2 and 3 have the minimal candidates, so their extensions are the normal explanations of $\bar{o}$. In this case the minimal conflict encoding is as complex as finding the extensions directly, but as the number of normal conditions gets larger, the minimal conflict encoding tends to be much more compact (see [de Kleer and Williams, 1987]).

If the normal explanations of an observation are not regular, then the method of comparing candidates of minimal explanations will not identify them. In the non-regular case, we must instead look at an expanded class of explanations, rather than just the minimal ones.

Definition 7 (Active explanation)
Any complement of a normal condition that has an explanation in a DCN is called an active condition. *An* active explanation *for an observation g is a minimal explanation for $\{g\} \cup W$, where W is any set of active conditions.*

That is, the active explanations for g are just the minimal explanations for g expanded by minimal sets of causes for some active conditions.

Fact 6 *Let Σ be the active explanations of g. The normal explanations of g are exactly the extensions of elements of Σ whose adjuncts are subset-minimal with respect to all other active candidates for the active explanations of g.*

Example 11 *Consider the basic xor circuit, with the addition that whenever c is abnormal it causes b to be abnormal: $\mathbf{ab}(c) \supset \mathbf{ab}(b) \in R$. Assuming $\bar{j}$ and i as initial conditions, we list the minimal conflicts and adjuncts for each of the active explanations of $\bar{o}$.*

	Active expl	Conflicts	Adjuncts
1.	$\bar{j}, i, \mathbf{ab}(c)$	$ok(b)$	$ok(c)$
		$ok(c)$	
2.	$\bar{j}, i, \mathbf{ab}(a), ok(c)$	$ok(a)$	$ok(a)$
3.	$\bar{j}, i, \mathbf{ab}(b), ok(c)$	$ok(b)$	$ok(b)$

The active explanations in this case are the minimal explanations although in general they need not be. The adjuncts can be computed from the conflicts; the adjunct of 1 does not contain $ok(b)$, because its complement is caused by $\mathbf{ab}(c)$, and so exempted. By comparing adjuncts we conclude that the extension of 1 (containing $ok(a)$) is normal, as well as the extensions of 2 and 3.

At this point we have enough results to form a proof method for finding the LIE's of an observation set O, assuming that the causal relation is finite, and all minimal conflicts are finite and computable.

1. Find the maximal subsets of O that have lenient explanations; call these the lenient subsets.
   $\Rightarrow$ Find minimal causal explanations for all subsets of O, and check whether they are consistent with $O \cup I$. Choose the maximal subsets of O that have such explanations.

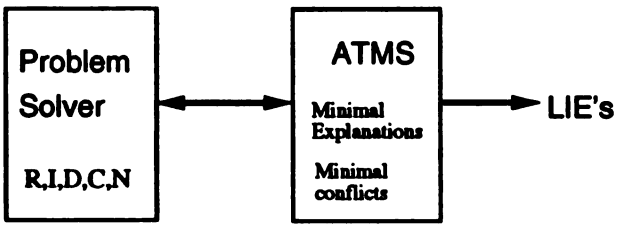


Figure 5: A Problem-Solving Architecture

2. Find the active explanations for these subsets of O.
   ⇒ Adjoin to O all the possible subsets of the active conditions, and find all minimal explanations for each.
3. Find the normal explanations among all explanations for the lenient subsets of O.
   ⇒ Find the conflicts of the active explanations of the previous step, and generate the adjuncts. The normal explanations are the extensions of the active explanations with minimal active candidates.
4. The ideal explanations are the normal explanations with minimal nonnormal causes.

3.2 ATMS implementation

The proof method just given can be implemented using a modification of the computational techniques available in the ATMS. Because the full language of the correlational theory I of DCNs is first-order, and the causal relation may be infinite, it is not possible in general to have a complete proof theory (as is also the case for normal default logic). Instead, the computation of LIE's can be phrased in terms of a dialogue between a problem solver and the ATMS [de Kleer, 1986], as in Figure 3.2.

The problem solver is an inference engine containing the DCN theory. It computes two kinds of structures and sends them to the ATMS. The definite clause causation relation is sent directly as an ATMS definite clause. These clauses are stored by the ATMS and used to compute the minimal explanations for all literals c present in the ATMS.

The ATMS keeps track of inconsistent sets of causes and normal conditions through the use of its NOGOOD mechanism. Whenever the problem solver finds a set of literals whose conjunction is inconsistent with I, it can send them to the ATMS as a NOGOOD. The *label* of a node g in the ATMS is the set of all minimal $A \subseteq C \cup N$ such that $A \models_R g$ and A is consistent with the NOGOODS. If the ATMS has the complete relation R, and the NOGOODS completely cover the inconsistent causes, then the label of g is the set of minimal explanations for g.

We need two additional structures: an observation set node, and an active condition set node. LIE's are found by computing the candidates of the labels of the conjunction of these two nodes.

We have successfully tried all of the problems in this paper. The computational properties are good, because the causal relation focuses the ATMS.

4 Relation to other approaches

There are two main representational differences between DCNs and model-based diagnosis methods based on Reiter's consistency technique [Reiter, 1987]. The first is that DCNs use an abductive method for generating explanations, while model-based methods use consistency with respect to the domain theory. The general relationship between these methods was pointed out in [Konolige, 1992]. From the theorems presented there, it is easy to show (as we have done in Fact 2) that if we represent the domain theory using only the correlational theory I, then excuses in DCNs correspond to diagnoses in the consistency-based approach.

On the other hand, the representation of causation available in DCNs leads to encodings of the domain, and explanations, that are not available in the model-based approach. In particular, we distinguish between strong and weak types of explanation: the cause of an observation versus an excuse for its negation. There is no counterpart to this in model-based diagnosis, since causal and correlational information is not distinguished in the domain theory. This has two benefits: we can use whatever information is available in partial fault models to find strong explanations, as in Example 5. And, without using any fault models or "physical impossibility" axioms, we can solve problems like the 3-bulbs example (8) using a combination of explanation and excuses. Making use of the role of causation in explanation yields a simpler representation of the domain.

We have also argued that preferences among explanations can be based on noting how causation and defaults interact, as in Example 5. Such preferences seem to follow commonsense reasoning based on causal knowledge. In model-based diagnosis, any assumptions about causation and defaults are implicit in the representation of components as being normal or abnormal, and the search for diagnoses as based on abnormal components. Such a view, we argue, is representationally restrictive, and does not give a deep enough analysis about how defaults interact. For example, although we can state relations among abnormalities in the domain (such as $ab(a) \supset ab(b)$), these relations do not necessarily lead to intuitively correct diagnoses in the consistency-based approach, because material implications within the framework are not treated as causal relations.

It should be pointed out that Reiter's original definition of diagnosis was biased towards system descriptions that only axiomatized the normal functioning of components. It was soon recognized that this definition would not work properly with partial fault models: if the fault models are not exhaustive, nothing useful can be inferred from a failing component (see [de Kleer and Williams, 1989]). This is strictly a consequence of the consistency-based approach; in an abductive approach, like that of DCNs, partial fault models are useful.

With exhaustive fault models, the consistency-based approach will produce useful diagnoses. However, one of the primary properties of diagnoses, that they are a parsimonious representation of all possible ways the system can fail, no longer holds. More recent work [de Kleer *et al.*, 1990] has defined the concept of *kernel diagnoses*, which in some cases can be a parsimonious representation. In this paper we have concentrated on defining the most normal explanation or excuse for an observation set. If we are interested in characterizing all excuses, then the ideas behind kernel diagnoses can be exploited in DCNs. However, this is not the case with explanations, since the asymmetry of the causal theory will undermine the this development.

4.1 Other approaches to explanation

Although we have concentrated on the application of DCNs to diagnosis, they provide a general framework for representing causation and explanation. Causation can be used as a unifying concept to understand various perspectives on diagnosis: excusing vs. explaining observations, correlation vs. causation, and the integration of normal conditions with explanatory causes. Although many of these issues have been dealt with separately in the literature, there have been few attempts to draw them together into a single framework, and the issues are often obscured by the formal or computational paradigm. There are many formal nonmonotonic systems that provide similar capabilities, although they are not phrased in terms of causation, e.g., Poole's THEORIST [Poole, 1988]. DCNs are distinguished by providing a coherent account of causation, correlation, and default conditions. Console et. al. [Console *et al.*, 1988] offer a logic-based translation of an expert system that uses causal knowledge, but since the translation is into a single first-order theory, they cannot model the same interaction of causation and defaults as in DCNs. Perhaps the closest system is Geffner's theory of causal and conditional reasoning [Geffner, 1989], which also takes causation as a primitive concept, and ties together explanation, defaults, and causation. He provides a complex but plausible formal account of these concepts, using a modal expression $C\alpha$ to represent "α is caused." Although the formalisms differ, there are many points of similarity between this work and his. Perhaps the major difference is that the roots of DCNs are default logic and abductive inference, and thus there are natural computational methods using the ATMS.

A good test of the DCN framework is the application to reasoning about events. We have started this task, and it appears that the problems of causation, explanation, and prediction in an event calculus can be treated within the DCN framework. The approach is similar to that of Shanahan [Shanahan, 1989], but the formal machinery is more general, and includes causation.

4.2 Some remarks about causation

Perhaps the weakest point of the DCN approach is that the theory of causation is not well developed. Since causation is treated as a proof-theoretic concept, there are some obvious problems (or, one might say opportunities) that arise. We discuss some of these here; a slightly more detailed treatment can be found in [Konolige, 1991].

First, there is a deliberate sloppiness about stating propositions in the causal relation. Most of the ones used in this paper are statements about a particular properties, e.g., the switch is closed or the light is on. But causation also involves events: "closing the switch caused the light to go on." We are trying to be as noncommittal as possible about the ontology of events and propositions, whether states of the world can be allowed as causes, how to specify the time of events, and so on. Any consistent defensible set of choices will do.

The second point is that the definite clauses of R specify *all* and *only* the propositions governing an effect. Closing the switch only turns on the bulbs if they are ok and the wires are intact. Of course, in any real-world situation there will be an inordinate number of such conditions, so any default causal theory will be relative to a set of background assumptions that do not enter into the theory. The choice of these assumptions is conventional.

It is important that only the relevant propositions participate in the causal relation. If we add an irrelevant proposition to the antecedent of a clause, the relation would still be useful in the sense that conjunction of the antecedents produces the desired effect, but it would be misleading in implying that all the antecedents were necessary. In producing explanations, minimal causal antecedents are required in the causal relation to ensure that explanations do not contain irrelevant propositions.

The role of primitive causes is to define the propositions over which, in some sense, we can exercise direct control. The point at which we choose to define primitive causes is partly a matter of convention. Often bodily movements are taken to be the ultimate primi-

tive causes, but this viewpoint is unnecessarily restrictive. Any well-defined event or condition that we can reliably bring about will suffice for a primitive cause, as long as the purpose of producing explanations is to give a set of conditions that account for the observed facts, and over which we have control.

One way to understand the causation relation R is as a provability relation. The provability relation is composed from individual inference steps combined into a tree; in the same way, the causation relation is specified by combining definite clause inference steps into a proof. Like classical provability, causation is monotonic:

$$\text{If } A \vdash_R c \text{ and } B \supset A, \text{ then } B \vdash_R c$$

and cumulative:

$$\text{If } A \vdash_R c \text{ and } B, c \vdash_R d, \text{ then } A, B \vdash_R d.$$

As we have stated, the important part of the causal relation is that it captures the functional dependence of the domain variables; this is the main difference between a causal relation and a merely correlational one. The asymmetry of causation is represented by the asymmetry of inference in a definite clause theory.

These remarks leave open the question of whether, in a particular instance, it is possible to have a causation relation that is symmetric for two propositions, or more generally to have one that is cyclic, containing a loop that leads from a proposition back to the same proposition. Other commitments may answer this question: for instance, assuming that causes always precede their effects in time forces the causal relation to be acyclic. The definite clause theory itself does not enforce any acyclic condition.

There are some further complications in defining a causal relation that we will mention here, without offering any definitive solutions. The first is that of inferred causation. We mentioned this briefly in proposing the definitional theory in Section 2.2. We use only a simple form of definitions to represent complements; any full-fledged theory of causation should at least take into account abstraction relations among propositions, e.g., "A 40-watt bulb is a type of bulb."

Another problem arises when our knowledge of the causation relation is partial. We have already remarked that we may only know a subset of the actual causation relation. Other kinds of uncertainty also exist. For example, suppose we know that dialing the number "911" connects one with either the police or the fire department, but we don't know which. The action of dialing 911 is completely determinate, it's just that we don't know the exact outcome. To express epistemic uncertainty of this kind, it is necessary to describe the causation relation in an appropriate language. If we let c stand for the action of dialing 911, d for calling the police, and e for calling the fire department, then our knowledge is expressed by the statement:

$$\text{Either } c \vdash_R d \text{ or } c \vdash_R e.$$

DCNs are not expressive enough to state this; a language that talks about causation, such as Geffner's [Geffner, 1989], would be necessary.

Acknowledgements

The research reported in this paper was supported by the Office of Naval Research under Contract No. N00014-89-C-0095.

I would like to thank Hector Geffner, Moises Goldszmidt and Ilkka Niemelä for valuable discussions.

References

[Console *et al.*, 1988] L. Console, D. Theseider Dupre, and P. Torasso. Abductive reasoning through direct deduction from completed domain models. In Z. W. Ras, editor, *Methodologies for Intelligent Systems 4*, pages 175–182. North-Holland, 1988.

[de Kleer and Williams, 1987] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[de Kleer and Williams, 1989] Johan de Kleer and Brian C. Williams. Diagnosis with behavioral modes. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Detroit, MI, 1989.

[de Kleer *et al.*, 1990] Johan de Kleer, Alan Mackworth, and Ray Reiter. Characterizing diagnoses. In *Proceedings of the Conference of the American Association of Artificial Intelligence*, Boston, MA, 1990.

[de Kleer, 1986] Johan de Kleer. An assumption-based truth maintenance system. *Artificial Intelligence*, 28:127–162, 1986.

[Friedrich *et al.*, 1990] Gerhard Friedrich, Georg Gotllob, and Wolfgang Nejdl. Physical impossibility instead of fault models. In *Proceedings of the Conference of the American Association of Artificial Intelligence*. MIT Press, 1990.

[Geffner, 1989] Hector Geffner. *Default Reasoning: Causal and Conditional Theories*. PhD thesis, Department of Computer Science, University of California at Los Angeles, 1989.

[Konolige, 1991] Kurt Konolige. What's happening: elements of commonsense causation. In *Proceedings of the International Conference on Cognitive Science*, San Sebastian, Spain, May 1991.

[Konolige, 1992] Kurt Konolige. Abduction vs. closure in causal theories. *Artificial Intelligence*, 53(2–3), 1992.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[Poole, 1988] David Poole. A methodology for using a default and abductive reasoning system. Technical report, Department of Computer Science, University of Waterloo, Waterloo, Ontario, 1988.

[Reiter, 1987] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32, 1987.

[Shanahan, 1989] Murray Shanahan. Prediction is deduction but explanation is abduction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Detroit, MI, 1989.

[Shoham, 1987] Yoav Shoham. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press, Cambridge, Massachusetss, 1987.

[Struss and Dressler, 1989] P. Struss and O. Dressler. Physical negation – integrating fault models into the general diagnostic engine. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1318–1323, Detroit, MI, 1989.

Focusing on Independent Diagnosis Problems

Hartmut Freitag
Siemens AG, ZFE BT SE 2
Otto-Hahn-Ring 6
D-8000 Munich 83, Germany
freitag@ztivax.zfe.siemens.de

Gerhard Friedrich
Christian Doppler Laboratory for Expert Systems
Technische Universität Wien, Paniglgasse 16
A-1040 Vienna, Austria
friedrich@vexpert.dbai.tuwien.ac.at

Abstract

This paper presents an approach which scales up current model-based diagnosis techniques to large, unstructured systems. It is based on the fact that it is sufficient to consider only a small part of a system in order to determine those components which cause its malfunction. By identifying minimal substructures of the system whose diagnoses are independent of the rest we reduce the number of diagnoses to be investigated and restrict behavior prediction to that part of the system's structure which is necessary to discriminate among competing diagnoses. Hence, our approach has the potential to save computational costs significantly and therefore extends the applicability of model-based techniques.

1 INTRODUCTION

Over the last decade research in model-based diagnosis (e.g. [de Kleer and Williams, 1989; Struss and Dressler, 1989; Friedrich and Nejdl, 1990]) has produced a large number of interesting and innovative ideas and approaches. Model-based diagnosis has reached a state, where industrial applications are feasible and show promising results because of the advantages of this method concerning correctness, completeness, and maintainability. However, one source which may prevent the application of model-based diagnosis is the size of the models. In many real world applications we have to deal with thousands of components. [de Kleer and Williams, 1990] deals with such systems by exploiting a priori failure probabilities in order to focus on the most probable diagnoses.

However, even the prediction of the system's behavior based on explicit models (which is the very basis of model-based diagnosis) can be too inefficient and costly. One approach to deal with structural complexity is to exploit a hierarchical structure [Hamscher, 1991] of a system in order to reduce the number of components. Many systems, however, do not exhibit a hierarchical structure (for instance power distribution networks as described in [Beschta *et al.*, 1992] or audio-routing systems as described in [Fleischanderl and Friedrich, 1991]).

In this paper we present an approach which focuses on relevant substructures of a device but does not rely on a hierarchical structure or on probabilistic information. The basic idea is as follows: Using symptom information we identify interesting parts of the system. We focus on these interesting parts by extending them to the smallest substructures of the system which are independent of the rest of the system. Independence in this case means that any further extension does not discriminate among the diagnoses of the independent substructure and therefore need not be considered. This decomposition depends on the system description *and* the current observations whereas hierarchical structures provide only a fixed decomposition.

Using this approach we perform behavior prediction only for independent substructures which include those components we are interested in. Since these substructures are normally only a fraction of the system's structure (e.g. see the example in the next section) there is the potential to save enormous computational costs. The approach considers different independent substructures separately and does not combine their diagnoses (since they are independent this is unnecessary), and therefore far fewer diagnoses have to be investigated.

The paper is organized as follows: the following section demonstrates the basic ideas of our approach by an example. Section 3 introduces the basic concepts. Section 4 presents the focusing strategy and the algorithm for structural focusing.

2 A MOTIVATING EXAMPLE

As an example we use a part of an audio switching matrix typically used in broadcasting stations for the flexible connection of studios, recording devices, etc. This matrix is configured according to the customers' needs and can include up to 1000 inputs and outputs and, thus may consist of up to 1,000,000 components.

For our purpose it is sufficient to use a 1 × 1000 matrix which consists of one input amplifier, 1000 output amplifiers and 1000 switches. For the sake of simplicity we represent an audio matrix by buffers and and-gates which logically produces the same behavior (Figure 1). Furthermore, we assume wires to be correct and therefore neglect them. Suppose we observe the following situation: the input signal is 1, the first and-gate gets 0, all other and-gates get 1 as value at their right input. The output of buffer $C5$ is measured to be 0, the outputs of all other buffers are 1.

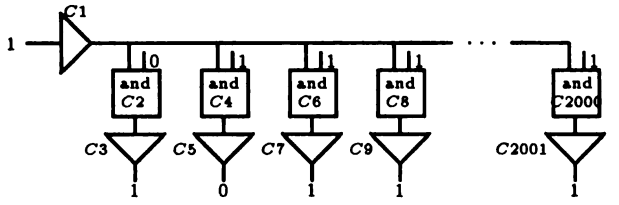


Figure 1: Logical Representation of a Small Audio Matrix

The misbehavior of such an audio switching system is recognized by continous monitoring (e.g. the operator recognizes a wrong or disturbed output or he/she gets an emergency call from some studios). The costs of such misbehavior are different and the operator has to fix the correct transmission to the most important outputs first.

2.1 FINDING DIAGNOSES

According to the behavior of buffers and and-gates one would expect the output of $C3$ to be 0 and all other outputs to be 1. In our example the operator monitoring the system detects two symptoms with different priority: the output of $C5$ is 0 (i.e. there is no audio signal) and the output of $C3$ is 1 (i.e. there is some audio signal although we expected none). The first symptom is more important than the second one and therefore has to be fixed first. It is important to note that these symptoms are detected by the operator or a monitoring system and not by the diagnostic system.

Applying GDE - a "standard" approach of model-based diagnosis [de Kleer and Williams, 1987] - to this problem we can derive the following disjunctions (in the GDE-terminology also called *conflicts*) by simulating the correct behavior of components and comparing the results to the observations: $ab(C2) \vee ab(C3)$ (because the output of $C2$ is independent of its left input), $ab(C1) \vee ab(C4) \vee ab(C5)$, $ab(C4) \vee ab(C5) \vee ab(C6) \vee ab(C7)$, ..., $ab(C4) \vee ab(C5) \vee ab(C2000) \vee ab(C2001)$.

Clearly, we are not interested in all possible diagnoses which can be generated from these conflicts, but prefer to consider only the most plausible diagnoses. The preference criterion in our example is defined by the number of suspected components: the set of preferred diagnoses is a minimal set of minimal diagnoses such that all non-preferred diagnoses suspect at least two components more than the worst preferred diagnosis, i.e. the minimum difference in suspected components between preferred and non-preferred diagnoses is 2.

Applying this preference criterion to our example yields the following diagnoses[1]: $[C2, C4]$, $[C2, C5]$, $[C3, C4]$, $[C3, C5]$. All other minimal diagnoses suspect at least four components and are therefore neglected. These conflicts and diagnoses were obtained by propagating all observed values through the whole device structure, which can cause enormous computational costs because of the large number of components. In the full application the total number of components may be greater than 1,000,000 depending on the configuration. However, even if we use a more sophisticated approach than GDE, like e.g. focused SHERLOCK [de Kleer and Williams, 1990], this propagation is required.

2.2 STRUCTURAL FOCUSING

The problem can be simplified, however, if we can exploit information about the importance of components to focus only on a part of the device (e.g. those very important parts which should be repaired as soon as possible). Tracing the faulty output of $C5$ (the most important symptom) back to its inputs by exploiting the IO-behavior of components we obtain the conflict $ab(C1) \vee ab(C4) \vee ab(C5)$ which provides $\{C1, C4, C5\}$ as an initial focused view of the device. Since at least one of these components is faulty (and therefore must be repaired in order to reestablish the purpose of the system), minimal preferred diagnoses for this *structure focus* are $[C1], [C4], [C5]$. [2]

Our diagnosis approach extends this structure focus by additional components only if they may provide information for discriminating between diagnoses for this structure focus. Different diagnoses predict different values at the output of $C1$: $[C1]$ predicts 0 (the observation can be propagated backwards by using only $\neg ab(C4)$ and $\neg ab(C5)$) and $[C4]$ and $[C5]$ predict 1 (using $\neg ab(C1)$). Thus, it may be possible to distinguish between diagnoses by propagating

[1] Components mentioned in the diagnosis are considered to be faulty. All other components are assumed to be correct.

[2] Components mentioned in the diagnoses are considered to be faulty. All other components in the *structure focus* are assumed to be correct.

these values using additional components. I.e. if additional inferences confirm or refute 0 or 1, some of these diagnoses are no longer appropriate. (Discrimination could also be achieved by directly observing the output of $C1$, but this output is not measurable). Therefore, we extend the structure focus by $C6$ (below we will see that it does not matter which of the components connected to $C1$ we choose). The diagnoses propose different values at the output of $C6$ ([$C4$] and [$C5$] predict 1, [$C1$] predicts 0) and therefore also $C7$ is considered which results in the additional conflict $ab(C4) \vee ab(C5) \vee ab(C6) \vee ab(C7)$.

The minimal preferred diagnoses [$C4$], [$C5$], [$C1, C6$], [$C1, C7$] again predict different values at the output of $C1$ ([$C4$], [$C5$] predict 1,[$C1, C6$], [$C1, C7$] predict 0) and therefore we have to extend the structure focus. Adding $C8$ and $C9$ results in the additional conflict $ab(C4) \vee ab(C5) \vee ab(C8) \vee ab(C9)$. This additional information is reflected by extending [$C1, C6$] and [$C1, C7$] by $C8$ or $C9$ which results in non-preferred diagnoses.

Neither of the two remaining minimal preferred diagnoses [$C4$], [$C5$] constrains the output of $C1$ which is the ***only*** connection of this substructure to the rest of the device. As an important consequence, no value restriction imposed by any other components on the output of $C1$ leads to a conflict including either $C4$ or $C5$. The structure $\{C1, C4, C5, C6, C7, C8, C9\}$ is therefore independent of the rest of the device with respect to the minimal preferred diagnoses. Each diagnosis of the rest of the device joined with a diagnosis of this independent structure is a diagnosis of the whole system.

Since our main goal is to re-establish the correct transmission to the output of $C5$ we can stop further deductions. Only an additional measurement between $C4$ and $C5$ will provide additional discrimination. Note, that even extending the initial structure focus $\{C1, C4, C5\}$ by $C2$ provides the same result: the output of $C2$ is independent of its left input and therefore any additional conflict which can be derived by propagating this output does not contain any of the components of the structure focus and thus can not distinguish between these diagnoses. This is also the reason why conflict $ab(C2) \vee ab(C3)$ need not be considered for resolving the first symptom.

In a similar way we can treat the second symptom (the output of $C3$ is 1), and find out that only an additional measurement between $C2$ and $C3$ can provide additional discrimination.

The strategy described above identifies interesting substructures of the device and extends them until they are independent of the rest of the device w.r.t. discrimination between minimal preferred diagnoses of this substructure. Compared to unfocused diagnosis this requires much less components (9 as opposed to 2001) to be considered. Additionally, since diagnoses of different independent subproblems are not combined we investigated only $9 + 2$ preferred diagnoses instead of $9 * 2$ ones.

3 BASIC CONCEPTS

3.1 PRELIMINARIES

We start by some elementary definitions following [de Kleer *et al.*, 1990]:

- A ***system*** is a triple $(SD, COMPS, OBS)$ where:
  - SD, the system description, is a set of first-order sentences;
  - $COMPS$, the system components, is a finite set of constants;
  - OBS, a set of observations, is a set of first-order sentences.
- A diagnosis is defined as a mode assignment for each component, i.e. for $D \subseteq COMPS$, $\Delta = \{ab(c) | c \in D\} \cup \{\neg ab(c) | c \in COMPS \setminus D\}$ is a ***diagnosis*** for $(SD, COMPS, OBS)$ ***iff*** $SD \cup OBS \cup \Delta$ is consistent.
- Furthermore, we define an ***ab-literal*** to be $ab(c)$ or $\neg ab(c)$ for some $c \in COMPS$. The components mentioned by a set of ab-literals (or a logical sentence) S are denoted by $comp(S)$.
- Finally, a ***conflict*** $CONFL$ of (SD, $COMPS$, OBS) is defined as a disjunction of ab-literals containing no complementary pair of ab-literals s.t. $SD \cup OBS \models CONFL$. A minimal conflict is a conflict such that no proper subclause is a conflict.

3.2 INDEPENDENT DIAGNOSIS PROBLEMS

In the example the independence of some substructure from the rest of the device was crucially dependent on the notion of ***discrimination*** between diagnoses. We can formally define this as follows:

Definition 1 (Discriminating Extension)
Let $(SD, COMPS, OBS)$ be a system. An extension $(COMPS', OBS')$ of this system is ***discriminating iff*** there are two diagnoses Δ_1, Δ_2 for $(SD, COMPS, OBS)$ and a (possibly empty) set of ab-literals EXT ($comp(EXT) \subseteq COMPS'$) such that

- $SD \cup OBS \cup OBS' \cup EXT \cup \Delta_1$ satisfiable
- $SD \cup OBS \cup OBS' \cup EXT \cup \Delta_2$ not satisfiable.

According to this definition an additional observation is discriminating if at least two diagnoses disagree about this observation. In contrast to [Reiter, 1987; McIlraith and Reiter, 1991] (which use only additional

observations) also an additional component C may be discriminating if at least two diagnoses disagree about mode assignments for this component, e.g. one diagnosis can be extended by $\neg ab(C)$, but not the other.

From this definition we can characterize the existence of a discriminating extension by the following lemma.

Lemma 1 There is no discriminating extension of some system $(SD, COMPS, OBS)$ *iff* for every extension $(COMPS', OBS')$ and every set of ab-literals EXT ($comp(EXT) \subseteq COMPS'$) the following holds: $SD \cup OBS \cup OBS' \cup EXT$ is either consistent with all diagnoses Δ of $(SD, COMPS, OBS)$ or none of them.

Using this conclusion we can prove the following theorem which provides the basis for characterizing device structures which are independent w.r.t. diagnoses.

Theorem 1
There is no discriminating extension $COMPS'$[3] of $IDP \subseteq COMPS$ ($COMPS' \subseteq COMPS \setminus IDP$) *iff* for all diagnoses Δ_i of (SD, IDP, OBS) and Δ_j of $(SD, COMPS \setminus IDP, OBS)$ follows that $\Delta_i \cup \Delta_j$ is a diagnosis for $(SD, COMPS, OBS)$.

The proof of this and all the following theorems can be found in the appendix.

According to this theorem a system can be split into two independent subsystems if there is such a subset IDP of components which has no discriminating extension. Every combination of diagnoses of the (smaller) independent subsystems is a diagnosis of the original system. Of course, every diagnosis of the original problem can be split into diagnoses for the two independent (smaller) subsystems by simply restricting the diagnoses to the components of the current subsystem. Note, that such a partition depends on the actual observations.

This motivates the definition of an independent diagnosis problem which uses the following notation.

Definition 2 ($\Delta \mid_D$) Let Δ be a set (conjunction) of ab-literals, $D \subseteq COMPS$ a set of components. Then $\Delta \mid_D$ is the set (conjunction) of all ab-literals in Δ which mention components of D. Let $DIAGS$ be a set of sets (conjunctions) of ab-literals then $DIAGS \mid_D$ denotes $\{\Delta \mid_D where \Delta \in DIAGS\}$.

Definition 3 (Independent Diagnosis Problem)
Let $(SD, COMPS, OBS)$ be a system. We call a subset $IDP \subseteq COMPS$ an *independent diagnosis problem iff* the following holds:

$\Delta \mid_{IDP}$ is a diagnosis of (SD, IDP, OBS) and $\Delta \mid_{COMPS \setminus IDP}$ is a diagnosis of (SD, $COMPS \setminus IDP$, OBS) *iff* Δ is a diagnosis of (SD, $COMPS$, OBS).

[3]Instead of $(COMPS', \emptyset)$ we often simply write $COMPS'$

A minimal independent diagnosis problem is an independent diagnosis problem which is minimal with respect to this property.

Characterizing independent diagnosis problems by means of discriminating extensions allows to focus diagnostic problem solving to those parts of a system we are interested in and to limit the investigation of the rest of the system with respect to discriminating power.

3.3 FOCUS OF ATTENTION

In many diagnosis problems, particularly when diagnosing large systems (e.g. an industrial plant), diagnosis is started with a certain goal, i.e. we only want to consider diagnoses with respect to a specific set of components. An example for such a set are those components which are necessary to (re-)establish a certain purpose of a system [Friedrich *et al.*, 1992; Friedrich and Nejdl, 1992]. In this case we are not interested in the detection of some other malfunctioning parts unless we can gain additional knowledge for our current problems.

Definition 4 (Focus of Attention) The focus of attention FOA is a subset of the system components $COMPS$.

In our example the focus of attention is the set of components which should ensure the correct transmission of a signal to the output of buffer $C5$, i.e. $\{C1, C4, C5\}$. Section 4.1 will show how to identify an appropriate focus of attention.

In order to exploit this focus of attention for limiting the efforts for behavioral prediction we can use independent diagnosis problems:

Remark 1 In order to discriminate between the diagnoses of (SD, FOA, OBS), i.e. the diagnoses of the focus of attention FOA, it is sufficient to extend this focus to an independent diagnosis problem IDP such that $FOA \subseteq IDP \subseteq COMPS$.

Consequently, for identifying the modes of components in the focus of attention FOA the smallest substructure of the system which has to be considered is the smallest independent diagnosis problem $IDP \supseteq FOA$ which includes all these components. Section 4.2 specifies an algorithm which incrementally expands FOA until such an independent diagnosis problem is found.

3.4 PREFERRED DIAGNOSES AND DISCRIMINATION

Since in most cases it is unacceptable to consider the set of *all* diagnoses, most diagnosis approaches introduce some kind of preference criterion to focus on some subset of diagnoses which we call *preferred diagnoses.*

Definition 5 (Preferred Diagnoses) Let $(SD, COMPS, OBS)$ be a system, $DIAGS$ be the set of diagnoses of $(SD, COMPS, OBS)$. The set of preferred diagnoses $PDIAGS$ is a subset of $DIAGS$.

Note, that we do not introduce any restriction on the preference criterion. E.g. it may be defined by using the number of suspected components (as we did in our example) or may be based on fault probabilities of components.

The purpose of a preference criterion is to focus the problem solving process on the probable cases, thus distinguishing between preferred and non-preferred diagnoses. However, we have to make sure that the actual diagnosis (e.g. the actual modes of the components) is included in the set of preferred diagnoses in case the diagnosis process stops in order to guarantee the correctness of the diagnosis process. Thus, we require that after the diagnosis process stops the set of preferred diagnoses only shrinks with additional information which we formally define as reliability of a preference criterion. A common stop criterion is that no further discrimination between preferred diagnoses is possible.

Definition 6 (Reliability of Preference) A preference criterion for $(SD, COMPS, OBS)$ is *reliable* w.r.t. discrimination *iff*
if there is no further discriminating extension of OBS and $COMPS$ with respect to the preferred diagnoses $PDIAGS$ of $(SD, COMPS, OBS)$
then for all possible extensions $(COMPS', OBS')$ the preferred diagnoses $PDIAGS' \mid_{COMPS}$ are a subset of $PDIAGS$. I.e. a non-preferred diagnosis does not become a preferred one by additional extensions.

There are two contrary objectives for designing a preference criterion. The preference criterion should be as selective as possible, in order to provide a useful reduction of the candidate space. On the other hand it should avoid the case where additional information turns a non-preferred diagnosis into a preferred one although further discrimination w.r.t. the preferred diagnoses is not possible. Then the likelihood of misdiagnosis is low.

E.g. in our example we stopped extending the structure focus because there was no possibility to discriminate w.r.t. the preferred diagnoses $[C4]$ and $[C5]$ whereas there would have been a possibility to discriminate between *all* diagnoses.

This also reflects what engineers do if they use additional measurement equipment (i.e. extending the set of components). They use redundant measurement devices to differentiate because these devices may fail. But at the point where such a failure is very unlikely, they stop to use further equipment although there is the possibility that all used measurement devices have failed.

3.5 INDEPENDENCE W.R.T. PREFERRED DIAGNOSES

Based on the reliability of preference criteria we can infer the two following corollaries:

Corollary 1 Provided that the preference criterion is reliable then the following equivalence holds: There is no discriminating extension $COMPS'$ of $IDP \subseteq COMPS$ where $(COMPS' \subseteq COMPS \setminus IDP)$ with respect to the preferred diagnoses of (SD, IDP, OBS) *iff* for all preferred diagnoses Δ_i of (SD, IDP, OBS) and diagnoses Δ_j of $(SD, COMPS \setminus IDP, OBS)$ follows that $\Delta_i \cup \Delta_j$ is a diagnosis for $(SD, COMPS, OBS)$.

Corollary 2 Let $(SD, COMPS, OBS)$ be a system, $IDP \subseteq COMPS$ an independent diagnosis problem, and the preference criterion be reliable. If Δ is a preferred diagnosis of $(SD, COMPS, OBS)$ then $\Delta \mid_{IDP}$ is a preferred diagnosis of (SD, IDP, OBS).

Stating it differently, combining the preferred diagnoses of (SD, IDP, OBS) and $(SD, COMPS \setminus IDP, OBS)$ yields a set of diagnoses of $(SD, COMPS, OBS)$ which is a superset of the set of preferred diagnoses of $(SD, COMPS, OBS)$.

4 STRATEGY

4.1 IDENTIFYING THE FOCUS OF ATTENTION

Our focusing strategy is based on the observation that diagnosis is mostly performed in cases where a malfunction of a physical system is evident, i.e. there is a set of measurements which indicates faulty behavior. Such malfunctioning is recognized by a user of the device or a monitoring system which is not identical with the diagnosis system in most cases. Typically, such a monitoring system uses models different to those of the diagnosis system due to the nature of its task, e.g. different abstractions and simplifications, different views according to physical aspects etc. [Struss, 1991].

The set of measurements supplied by the monitoring system forms a symptom. The task of diagnosis is to point out the causes of this symptom. Our system uses symptoms to generate a first set of conflicts. This can be supported by causal dependency information and is easy to compute if the system and its components have definite IO-behavior. In the case where no initial symptom is supplied and no causal information can be exploited, we start the simulation until a conflict is detected. All components included in these conflicts form the initial focus of attention. Clearly, there may be also more sophisticated, domain-dependent strategies to identify the initial focus of attention.

4.2 IDENTIFYING INDEPENDENT DIAGNOSIS PROBLEMS

Having determined an appropriate focus of attention our goal is to find the smallest independent diagnosis problem which includes this focus of attention.

In the following we use the term *structure focus SF* to denote the set of components we currently consider.

Definition 7 (Structure Focus)
Let $(SD, COMPS, OBS)$ be a system. A *structure focus* SF is a set of components: $SF \subseteq COMPS$.

Furthermore, we assume that the behavior of a component is described by a set of constraints and constraints are connected by shared variables.

Let $vars(D), D \subseteq COMPS$ be the set of variables related to all components in D. Let $border(SF) = vars(COMPS \setminus SF) \cap vars(SF)$ be the set of variables which establish a connection between the components in the structure focus and the components outside the structure focus.

The algorithm for an incremental extension of the structure focus (which initially consists of the focus of attention) then looks as follows:

Algorithm 1 Structural Focusing

1. $SF \leftarrow FOA$
2. $VARS \leftarrow border(SF)$
3. while $VARS \neq \emptyset$ do
   $x \leftarrow$ select one variable of $VARS$
   $VARS \leftarrow VARS - x$
   if *discrimination-possible-p*$(x, SF, DIAGS)$
   then
      $C \leftarrow$ select one component connected to x
      $SF \leftarrow SF \cup \{C\}$
      $DIAGS' \leftarrow$ run-diagnosis(SD, OBS, SF)
      if $DIAGS = DIAGS'$
      then
         $VARS \leftarrow VARS \cup$
         $(vars(\{C\}) \cap vars(COMPS \setminus SF))$
      else
         $DIAGS \leftarrow DIAGS'$
         goto 2.
4. STOP ;;; SF is an independent diagnosis problem

The structure focus is intially set to the focus of attention and is then extended by a component connected to some border variable if this extension discriminates between the current diagnoses of the structure focus (in this case *discrimination-possible-p*$(x, SF, DIAGS)$ is true). Diagnoses of the extended structure focus are computed by *run-diagnosis*. In case this changes the current set of diagnoses we have to re-consider all border variables. This is done until the extension of SF by any connected component does not allow to discriminate between the current diagnoses.

The important decision in this algorithm is to find out whether an extension of SF may discriminate between diagnoses. Discrimination can only be achieved by additional conflicts which eliminate some diagnoses or extend them by some additional components.

Remark 2 Let Δ be a consistent set of ab-literals (representing candidates for diagnoses), $CONFL$ a new conflict. Then there are three different cases to be distinguished:

1. $\Delta \cup \{CONFL\}$ is satisfiable.
   (a) $\Delta \models CONFL$.
   (i.e. $CONFL$ is already covered by Δ.)
   (b) $\Delta \not\models CONFL$.
   (i.e. Δ has to be extended.)
2. $\Delta \cup \{CONFL\}$ is not satisfiable.
   (i.e. Δ has to be eliminated.)

From the first case we see that a new conflict which contains at least one ab-literal with the same sign from each diagnosis is entailed by all diagnoses and does not change the set of diangoses. Thus, a conflict $CONFL$ where $comp(CONFL) \subseteq \bigcup_{\Delta \in DIAGS} comp(\Delta)$ is already covered, where $DIAGS$ is the set of all diagnoses of the current SF.

A conflict which does not mention any component of the current SF is not entailed by any diagnosis of SF, but is consistent with all these diagnoses and therefore the second case applies for every diagnosis. Thus, a conflict $CONFL$ where $comp(CONFL) \cap \bigcup_{\Delta \in DIAGS} comp(\Delta) = \emptyset$ contributes to another independent diagnosis problem.

In the last case Δ has to be eliminated because it cannot resolve the new conflict.

This motivates the following definition:

Definition 8 (Discriminating Conflict) Let $DIAGS$ be a set of diagnoses of a structure focus SF. A conflict $CONFL$ is discriminating w.r.t. $DIAGS$ *iff*

- $\exists \Delta_1 \in DIAGS$ s.t. $\Delta_1 \models CONFL$ and
- $\exists \Delta_2 \in DIAGS$ s.t. $\Delta_2 \not\models CONFL$

The following theorem relates discriminating conflicts to discriminating extensions (and thus also to independent diagnosis problems).

Theorem 2 There is a discriminating extension $COMPS'$ of $SF \subseteq COMPS$ where $COMPS' \subseteq COMPS \setminus SF$ *iff* there is a discriminating minimal conflict of SF.

Thus, in order to determine whether *discrimination-possible-p*$(x, SF, DIAGS)$ is true for some variable x we have to find out whether discriminating minimal conflicts are possible.

4.3 DETECTING POSSIBLE DISCRIMINATING MINIMAL CONFLICTS

Deciding whether a discriminating minimal conflict is possible at some variable x is done by a kind of look-ahead procedure which is quite similar to the procedure used to assess the results of potential observations. Using constraints (possibly expressed by rules) describing the behavior of components, the predictive engine (i.e. the component of the diagnosis system which is responsible for simulating the behavior of the system to be diagnosed) derives value restrictions for variables. A value restriction $VR(x)$ for some variable x is a maximal subset of its domain $DOM(x)$ s.t. for each assignment $x = v$ where $v \in VR(x)$ the constraint network is satisfiable. Let CS be a constraint network then $CS \models \bigvee_{v \in VR(x)} x = v$ where no proper subdisjunct is implied by CS.

A discrepancy which causes the detection of a conflict occurs if the predictive engine derives two value restrictions for a variable which are incompatible because they are disjoint.

The decision procedure has to determine whether a conflict which can be detected by a discrepancy potentially occurring at x would be discriminating.

Corollary 3 A discriminating minimal conflict exists w.r.t. the current structure focus SF *iff* there is a set of ab-literals EXT ($comp(EXT) \subseteq COMPS \setminus SF$), two diagnoses Δ_1, Δ_2 of (SD, SF, OBS) and a variable x s.t. $VR(x) = \emptyset$ for $SD \cup OBS \cup \Delta_1 \cup EXT$ and $VR(x) \neq \emptyset$ for $SD \cup OBS \cup \Delta_2 \cup EXT$.

Proof: This corollary corresponds to Theorem 2 by making the condition for the existence of a discriminating extension of ab-literals explicit. □

For deciding if a discriminating minimal conflict is possible without investigating other components than those included in the current structure focus, the following condition has to be checked:

Condition 1 A conflict indicated by a potential discrepancy at variable x may be discriminating and minimal w.r.t. the current structure focus SF and diagnoses $DIAGS$ of (SD, SF, OBS) *iff*
$\exists \Delta_1, \Delta_2 \in DIAGS$ and a set of value assignments for a subset SUB of the border variables $VA = \{x_i = v_i | x_i \in SUB \subseteq border(SF) - x, v_i \in DOM(x_i)\}$ s.t. the value restriction $VR_1(x)$ induced by Δ_1, VA, SD, OBS and $VR_2(x)$ induced by Δ_2, VA, SD, OBS are incompatible.

A discriminating minimal conflict is possible whenever there is a value assignment to the variables at the border which invalidates one diagnosis of the structure focus but is consistent with another one.

Condition 1 is also sufficient: if the condition is not met then there is no discriminating minimal conflict. This means that the procedure basically has to check whether diagnoses predict or do not predict a particular value restriction.

The following remark is a special case of condition 1.

Remark 3 A sufficient condition that no discriminating minimal conflict is possible is:

all diagnoses of (SD, SF, OBS) predict a specific value for all $x \in border(SF)$ and they agree on the value for each x.

Example 1 Reconsider our example. There are two preferred diagnoses $[C4]$ and $[C5]$ of the structure focus $C1, C4, C5, \ldots, C9$ both predicting 1 at the output of $C1$. Note, that the output of $C1$ is the only border variable and therefore we are allowed to stop the extension of the structure focus. $C1, C4, C5, \ldots, C9$ is an independent diagnosis problem.

Using the notion of prime implicates we can rephrase this condition corresponding to results in [McIlraith and Reiter, 1991] for discrimination between diagnoses using additional observations.

Definition 9 (Prime Implicate) A prime implicate of a propositional formula Σ is a clause C such that

1. $\Sigma \models C$
2. For no proper subclause C' of C does $\Sigma \models C'$

Condition 2 Let $DIAGS$ be the set of diagnoses of (SD, SF, OBS).

A discriminating minimal conflict is possible at x *iff* there are diagnoses $\Delta_1, \Delta_2 \in DIAGS$ and a value assignment $x = v$ and value assignments $VA = \{x_i = v_i | x_i \in border(SF) - x, v_i \in DOM(x_i)\}$ and prime implicates $\neg H' \vee x = v$ and $\neg H'' \vee x \neq v$ *of* $SD \cup OBS$ where H' and H'' are subconjuncts of $H_1 = \Delta_1 \wedge \bigwedge_{x_i = v_i \in VA} x_i = v_i$ and $H_2 = \Delta_2 \wedge \bigwedge_{x_i = v_i \in VA} x_i = v_i$.

This alternative characterization also provides a basis for an implementation of our procedure: As shown in [Reiter and de Kleer, 1987] the principal task of an assumption-based truth maintenance system (ATMS) [de Kleer, 1986] is the computation of all prime implicates of a background theory. Therefore, in cases where the predictive engine uses an ATMS to record its inference steps, we can simply exploit the labels computed by the ATMS to decide whether a discriminating minimal conflict is potentially possible.

4.4 DEALING WITH INCOMPLETENESS

We showed in the previous section that the evaluation of Condition 1 is very efficient if every diagnosis of a

structure focus predicts a unique value for all border variables of the structure focus. We only have to compare the values of the border variables.

However, if some diagnoses do not predict a value for some border variables then the costs of deciding if no discriminating minimal conflict exists for such a variable have to be compared to the costs of extending the structure focus. This decision is related to preprocessing techniques employed in constraint networks (e.g. [Dechter and Pearl, 1989]). Additionally, there is a close relation between deciding whether there may exist a discriminating minimal conflict and the interchangeability of values ([Freuder, 1991]) of the border variables with respect to the diagnoses of the structure focus.

One strategy to eliminate border variables where no value is predicted is to show that all constraints which link a variable to other variables of the actual structure focus are redundant. [Dechter and Dechter, 1987] deals with removing redundancies in constraint networks showing the benefits for solving constraint satisfaction problems.

In the following we state several conditions for the redundancy of causal constraints (defined in [Dechter and Pearl, 1991], p. 1169) of a constraint network (e.g. the actual structure focus) SF. Note, that any assignment to the input variables of a causal constraint is legal.

1. A causal constraint C is redundant if the output variables of C are not connected to any other constraint (Note, that observations are constraints).
2. Let C be a constraint where values are predicted at the outputs of C and $I' \subseteq I$ is the set of input variables which are connected to other constraints and where no value is predicted. If any value assignments to input variables I' are consistent with the causal constraint C then C is redundant.
3. Let C be a causal constraint where values are predicted at the outputs of C. If we use a rule-based approach to model functional constraints, then it is sufficient to check if simulation rules [Davis, 1984], (i.e. rules using some input variables to deduce the value of an output) are applicable s.t. a value is assigned to each output, i.e. these rules are consistent with the value assigned to the outputs.

The conditions described for removing redundancy are efficient to evaluate. These conditions introduce a linear time overhead in the number of constraints. Note, that removing constraints is monotonic, i.e. we can reuse the pruned structure focus if it has to be extended.

Although these conditions are computationally very easy to evaluate they are powerful for limiting the extension of the structure focus. In the following example we demonstrate how the simulation of n-bit ripple carry adder (introduced by [de Kleer and Williams, 1990]) can be limited to four components instead of $5 \times n$ components.

Consider an n-bit ripple carry adder depicted in Figure 2. If a component is behaving abnormally, then the output is either stuck at zero or stuck at one. Furthermore, the preference criterion focuses on minimal diagnoses, e.g. non-minimal diagnoses are not considered.

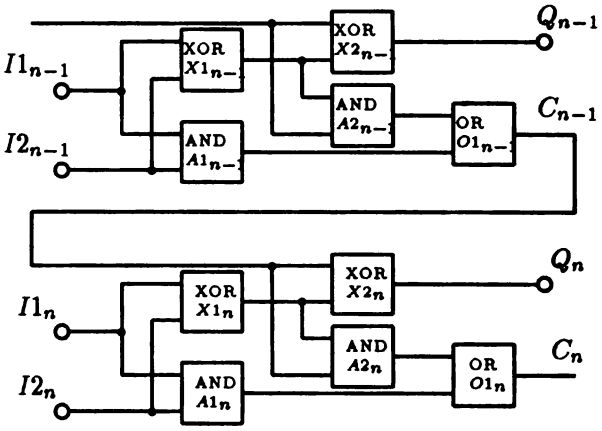


Figure 2: n-bit Ripple Carry Adder

Suppose that all inputs except the most significant bit are zero, i.e. $I1 = 2^{n-1}$ and $I2 = 2^{n-1}$, but the output Q is 0 and also C_n is 0. Using a monitoring model one can easily find the wrong output bits:

100000...	observed
+ 100000...	observed
1000000...	predicted
0000000...	observed

Using signal tracing we identify components $A1_n$, $A2_n$, $O1_n$, $X1_n$ as our focus of attention and initial structure focus. At least one of these components has to be abnormal. Running the diagnostic engine on this structure focus leads to two minimal diagnoses $[A1_n]$ and $[O1_n]$. Note, that there are two variables included in the border of the structure focus, i.e. both inputs of $A2_n$. We call the input connected to the output of $X1_n$ $in1$ and the other $in2$. Both diagnoses agree on the value of $in1$ which is 0. However, both diagnoses do not predict a value for $in2$. Since a simulation rule of $A2_n$ is applicable, constraint $A2_n$ is redundant and can be deleted, i.e. $in2$ is masked. It follows that $in2$ can be eliminated from the border. Since all diagnoses agree on the values of the remaining border variables, the actual structure focus is an independent diagnosis problem. After observing the output of $A1_n$, where

the only discriminating observations are, the diagnosis process stops.

Combining the monitoring model and our technique for discrimination-driven focusing we limited the simulation to four components needed in the diagnosis phase.

5 CONCLUSIONS

We have presented an approach which deals with structural complexity using the concept of independent diagnosis problems. By focusing on minimal independent diagnosis problems and considering different such problems separately we can reduce the number of components which have to be considered for behavior prediction and the number of candidates for diagnosis to be investigated. The algorithm for maintaining the structural focus uses a decision procedure which can exploit ATMS-labels in ATMS-based diagnosis systems like GDE$^+$ [Struss and Dressler, 1989] or SHERLOCK [de Kleer and Williams, 1989] and thus reduces the additional overhead in such systems. Consequently, our approach significantly broadens the applicability of model-based diagnosis.

Acknowledgements

The content of this paper has benefited from discussions with Anton Beschta, Philippe Dague, Johan de Kleer, Oskar Dressler, Walter Hamscher, Michael Montag, Ray Reiter, Peter Struss. The first author especially thanks Georg Gottlob for inviting him to a research visit at the Technische Universität Wien. The second author especially thanks Peter Struss for inviting him to a research visit at SIEMENS.

This work was supported in part by BMFT (ITW9001 A9), CEC (P5143,3249) and the Christian Doppler Laboratory for Expert Systems of the Austrian Industries.

APPENDIX: PROOFS

Theorem 1
There is no discriminating extension $COMPS'$ of $IDP \subseteq COMPS$ ($COMPS' \subseteq COMPS \setminus IDP$) *iff* for all diagnoses Δ_i of (SD, IDP, OBS) and Δ_j of $(SD, COMPS \setminus IDP, OBS)$ follows that $\Delta_i \cup \Delta_j$ is a diagnosis for $(SD, COMPS, OBS)$.

Proof: ($\Rightarrow$) Assume that there is no discriminating extension of IDP and there is a diagnosis Δ_i of (SD, IDP, OBS) and a diagnosis Δ_j of $(SD, COMPS \setminus IDP, OBS)$ such that $\Delta_i \cup \Delta_j$ is not a diagnosis for $(SD, COMPS, OBS)$. Therefore, $\Delta_i \cup \Delta_j \cup SD \cup OBS$ is inconsistent.

Because there is no discriminating extension of IDP it follows that all diagnoses of IDP are inconsistent with $SD \cup OBS \cup \Delta_j$. Therefore, $SD \cup OBS \models \neg\Delta_j$ which is a contradiction to the fact that Δ_j is a diagnosis of $(SD, COMPS \setminus IDP, OBS)$.

($\Leftarrow$) Assume that for all diagnoses Δ_i of (SD, IDP, OBS) and Δ_j of $(SD, COMPS \setminus IDP, OBS)$ follows that $\Delta_i \cup \Delta_j$ is a diagnosis for (SD, $COMPS$, OBS) but there is a discriminating extension $COMPS'$ of IDP.

In this case there are two diagnoses Δ_1, Δ_2 of (SD, IDP, OBS) such that

- $SD \cup OBS \cup EXT \cup \Delta_1$ satisfiable
- $SD \cup OBS \cup EXT \cup \Delta_2$ not satisfiable

with $comp(EXT) \subseteq COMPS'$.

It follows that $SD \cup OBS \cup EXT$ is satisfiable and therefore there is a diagnosis $\Delta_j \supseteq EXT$ of $(SD, COMPS \setminus IDP, OBS)$. However, $\Delta_2 \cup \Delta_j$ is not a diagnosis for $(SD, COMPS, OBS)$ which is a contradiction to the presupposition. □

Theorem 2 There is a discriminating extension $COMPS'$ of $SF \subseteq COMPS$ where $COMPS' \subseteq COMPS \setminus SF$ *iff* there is a discriminating minimal conflict of SF.

Proof: ($\Rightarrow$) Since a discriminating extension $COMPS'$ exists, there are two diagnoses Δ_1, Δ_2 for (SD, SF, OBS) and a set of ab-literals EXT ($comp(EXT) \subseteq COMPS'$) such that

- $SD \cup OBS \cup EXT \cup \Delta_1$ satisfiable
- $SD \cup OBS \cup EXT \cup \Delta_2$ not satisfiable.

Therefore, $\bigvee_{L \in \Delta_2 \cup EXT} \neg L$ is a conflict set and $\Delta_2 \not\models \bigvee_{L \in \Delta_2 \cup EXT} \neg L$. Note, that Δ_1 and Δ_2 contain the same ab-literals but at least one sign of an ab-literal contained in both diagnoses differs. Furthermore $\bigvee_{L \in EXT} \neg L$ is not a conflict since $SD \cup OBS \cup EXT \cup \Delta_1$ is satisfiable. Therefore every consistent extension of EXT by some ab-literals $SUB \subseteq \{\neg ab(c), ab(c) | c \in SF\}$ which leads to an inconsistency with $SD \cup OBS$ has to contain at least one ab-literal of Δ_1 with opposite sign. Therefore the negation of $EXT \cup SUB$ is a conflict set which is implied by Δ_1. Since every conflict of this form is implied by Δ_1 also the minimal ones are implied.

($\Leftarrow$) Let $DIAGS$ be a set of diagnoses of a structure focus SF and $CONFL$ be a discriminating minimal conflict w.r.t. $DIAGS$. Therefore, there are at least two diagnoses in $DIAGS$ such that

- $\Delta_1 \models CONFL$ and
- $\Delta_2 \not\models CONFL$

Δ_2 is a diagnosis for SF and therefore implies all conflicts which mention only components of SF. It follows, $CONFL$ is a consistent disjunction of ab-literals containing some literals not mentioned in Δ_2. We call $EXT = \neg CONFL\,|_{COMPS \setminus SF}$ which consists of ab-literals not mentioning components of SF and $SUB = \neg CONFL\,|_{SF}$ including the ab-literals mentioning components of SF.

Since Δ_2 does not imply $CONFL$, ab-literals of Δ_2 and SUB have same signs. Therefore SUB is a subset of Δ_2. This subset is not empty since Δ_1 implies $CONFL$ and therefore some ab-literals of Δ_1 are contained in SUB and have opposite signs.

Since $CONFL$ is a conflict $SD \cup OBS \cup \{\neg CONFL\}$ is inconsistent. Since $\neg CONFL$ is a subconjunct of $\Delta_2 \cup \{\neg EXT\}$ there exists an extension to SF s.t. $SD \cup OBS \cup \Delta_2 \cup \{\neg EXT\}$ is not satisfiable.

Finally, we have to show that there exists a diagnosis Δ_1 of (SD, SF, OBS) s.t. $SD \cup OBS \cup \Delta_1 \cup \{\neg EXT\}$ is satisfiable. Suppose this set of sentences is unsatisfiable, i.e. for every diagnosis Δ_i of (SD, SF, OBS) $SD \cup OBS \cup \Delta_i \cup \{\neg EXT\}$ is unsatisfiable. Therefore, EXT itself is a conflict set. It follows that EXT is a strict subdisjunct of $CONFL$ because some ab-literals of $CONFL$ have to mention components of SF. This is a contradiction to the fact that $CONFL$ is a minimal conflict set. □

Corollary 1 Provided that the preference criterion is reliable then the following equivalence holds: There is no discriminating extension $COMPS'$ of $IDP \subseteq COMPS$ where $(COMPS' \subseteq COMPS \setminus IDP)$ with respect to the preferred diagnoses of (SD, IDP, OBS) *iff* for all preferred diagnoses Δ_i of (SD, IDP, OBS) and diagnoses Δ_j of $(SD, COMPS \setminus IDP, OBS)$ follows that $\Delta_i \cup \Delta_j$ is a diagnosis for $(SD, COMPS, OBS)$.

Proof: ($\Rightarrow$) Assume that there is no discriminating extension of IDP and there is a diagnosis Δ_i of (SD, IDP, OBS) and a diagnosis Δ_j of $(SD, COMPS \setminus IDP, OBS)$ such that $\Delta_i \cup \Delta_j$ is not a diagnosis for $(SD, COMPS, OBS)$. Therefore, $\Delta_i \cup \Delta_j \cup SD \cup OBS$ is inconsistent.

Because there is no discriminating extension of IDP it follows that all preferred diagnoses of IDP are inconsistent with $SD \cup OBS \cup \Delta_j$. Since Δ_j is a diagnosis for $(SD, COMPS \setminus IDP, OBS)$ it follows that $SD \cup OBS$ is consistent and therefore a diagnosis for $(SD, COMPS, OBS)$ exists. It follows that a diagnosis for (SD, IDP, OBS) exists which is not in the set of preferred diagnoses. Therefore, there is an extension to (SD, IDP, OBS) s.t. a non-preferred diagnosis becomes a preferred one. This contradicts the reliability of the preference criterion.

($\Leftarrow$)[4] Assume that for all preferred diagnoses Δ_i of (SD, IDP, OBS) and all diagnoses Δ_j of $(SD, COMPS \setminus IDP, OBS)$ follows that $\Delta_i \cup \Delta_j$ is a diagnosis for $(SD, COMPS, OBS)$ but there is a discriminating extension $COMPS'$ of IDP.

In this case there are two preferred diagnoses Δ_1, Δ_2 of (SD, IDP, OBS) such that

- $SD \cup OBS \cup EXT \cup \Delta_1$ satisfiable
- $SD \cup OBS \cup EXT \cup \Delta_2$ not satisfiable

with $comp(EXT) \subseteq COMPS'$.

It follows that $SD \cup OBS \cup EXT$ is satisfiable and therefore there is a diagnosis $\Delta_j \supseteq EXT$ of $(SD, COMPS \setminus IDP, OBS)$. However, $\Delta_2 \cup \Delta_j$ is not a diagnosis for $(SD, COMPS, OBS)$ which is a contradiction to the presupposition. □

Corollary 2 Let $(SD, COMPS, OBS)$ be a system, $IDP \subseteq COMPS$ an independent diagnosis problem, and the preference criterion be reliable.

If Δ is a preferred diagnosis of $(SD, COMPS, OBS)$ then $\Delta\,|_{IDP}$ is a preferred diagnosis of (SD, IDP, OBS).

Proof: Suppose Δ is a preferred diagnosis of $(SD, COMPS, OBS)$ but $\Delta\,|_{IDP}$ is not a preferred diagnosis of (SD, IDP, OBS). In this case $COMPS \setminus IDP$ is an extension to (SD, IDP, OBS) s.t. the preferred diagnoses $PDIAGS\,|_{IDP}$ is not a subset of the preferred diagnoses of (SD, IDP, OBS) which is a contradiction to the reliability of the preference criterion. □

References

[Beschta *et al.*, 1992] Beschta, Anton; Dressler, Oskar; Freitag, Hartmut; Montag, Michael; and Struss, Peter 1992. A model-based approach to fault localization in power transmission networks. *Intelligent Systems Engineering.*

[Davis, 1984] Davis, Randall 1984. Diagnostic reasoning based on structure and behaviour. *Artificial Intelligence* 24:347–410.

[de Kleer and Williams, 1987] de Kleer, Johan and Williams, Brian C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32:97–130.

[de Kleer and Williams, 1989] de Kleer, Johan and Williams, Brian C. 1989. Diagnosis with behavioral modes. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Detroit. Morgan Kaufmann Publishers, Inc. 1324–1330.

[4] Analog to proof of Theorem 1.

[de Kleer and Williams, 1990] de Kleer, Johan and Williams, Brian C. 1990. Focusing the diagnosis engine. In *First International Workshop on Principles of Diagnosis*, Stanford.

[de Kleer *et al.*, 1990] de Kleer, Johan; Mackworth, Alan K.; and Reiter, Raymond 1990. Characterizing diagnoses. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Boston. Morgan Kaufmann Publishers, Inc. 324–330.

[de Kleer, 1986] de Kleer, Johan 1986. An assumption-based TMS. *Artificial Intelligence* 28:127–162.

[Dechter and Dechter, 1987] Dechter, Avi and Dechter, Rina 1987. Removing redundancies in constraint networks. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Seattle.

[Dechter and Pearl, 1989] Dechter, Rina and Pearl, Judea 1989. Tree clustering for constraint networks. *Artificial Intelligence* 38:353–366.

[Dechter and Pearl, 1991] Dechter, Rina and Pearl, Judea 1991. Directed constraint networks: A relational framework for causal modeling. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Sydney, Australia. Morgan Kaufmann Publishers, Inc. 1164–1170.

[Fleischanderl and Friedrich, 1991] Fleischanderl, Gerhard and Friedrich, Gerhard 1991. The artex project and the lessons learned for diagnosis. In *European Conference on Industrial Applications of Knowledge-Based Diagnosis*, Milano.

[Freuder, 1991] Freuder, Eugene C. 1991. Eliminating interchangeable values in constraint satisfaction problems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Anaheim, CA. 227–233.

[Friedrich and Nejdl, 1990] Friedrich, Gerhard and Nejdl, Wolfgang 1990. MOMO — Model-based diagnosis for everybody. In *Proceedings of the IEEE Conference on Artificial Intelligence Applications (CAIA)*, Santa Barbara.

[Friedrich and Nejdl, 1992] Friedrich, Gerhard and Nejdl, Wolfgang 1992. Choosing observations and actions in model-based diagnosis-repair systems. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA.

[Friedrich *et al.*, 1992] Friedrich, Gerhard; Gottlob, Georg; and Nejdl, Wolfgang 1992. Formalizing the repair process. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, Vienna. Also appeared in the Proceedings of the Second International Workshop on Principles of Diagnosis, Milano, 1991.

[Hamscher, 1991] Hamscher, Walter 1991. Modeling digital circuits for troubleshooting. *Artificial Intelligence* 51:223 - 271.

[McIlraith and Reiter, 1991] McIlraith, Sheila and Reiter, Raymond 1991. On experiments for hypothetical reasoning. In *Proceedings of the Second International Workshop on Principles of Diagnosis*, Milano.

[Reiter and de Kleer, 1987] Reiter, Raymond and de Kleer, Johan 1987. Foundations of assumption-based truth maintenance systems: Preliminary report. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Seattle. 183–188.

[Reiter, 1987] Reiter, Raymond 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32:57–95.

[Struss and Dressler, 1989] Struss, Peter and Dressler, Oskar 1989. Physical negation — Integrating fault models into the general diagnostic engine. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Detroit. Morgan Kaufmann Publishers, Inc. 1318–1323.

[Struss, 1991] Struss, Peter 1991. A theory of model simplification and abstraction for diagnosis. Presentation at the 5th International Workshop on Qualitative Reasoning QR-91.

A Minimality Maintenance System

Olivier Raiman and Johan de Kleer
Xerox Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304 USA
email: raiman@xerox.com, dekleer@xerox.com

Abstract

The functionality of Assumption-Based Truth Maintenance Systems (ATMS's) have made them a widely used tool in Artificial Intelligence. But many tasks such as commonsense reasoning, diagnosis and dynamic constraint satisfaction require a form of non-monotonic reasoning beyond the scope of the ATMS. The Minimality Maintenance System (MMS) has the same functionality and supports exactly the same set of transactions as the ATMS except that the underlying theory is circumscribed according to some circumscriptive policy. To build a MMS we generalize the notion of a support, which is at the foundation of ATMSs, to that of circumscriptive supports. The circumscriptive supports can be obtained by constructing the prime implicates of the originally supplied set of formulae augmented by formulae which characterize the circumscriptive policy. The MMS has been completely implemented and tested on many examples.

1 Introduction

Many AI implementations perform non-monotonic reasoning of some kind. Truth maintenance systems evolved, in part, to facilitate the implementation of problem-solvers which needed to perform non-monotonic reasoning. A key advantage of truth maintenance systems is that they precisely formalize a key inferential component with no requirement that the rest of the task be formalized. This property is responsible for the wide use of TMSs in AI because most actual implementations are not embedded within a formal or logical framework.

Unfortunately, most TMSs still do not support the full scope of non-monotonic reasoning required in implementations. Hence, many implementations still, with the help of some form of TMS, perform non-monotonic reasoning themselves. (Some of the first TMSs were non-monotonic (e.g., [10]) but their semantics have often been murky and they have not found widespread use.)

2 Task

For many applications an Assumption-Based Truth Maintenance System (ATMS) [2; 26] provides the necessary functionality. The ATMS is provided a sequence of propositional clauses interspersed with queries. A query takes the form of a literal l. The ATMS is expected to respond with every shortest clause S for which $S \vee l$ is a logical consequence, but S is not a logical consequence, of the clauses transmitted thus far to the ATMS. Two generic examples of ATMS use are in search and abduction. In abduction tasks, $\neg S$ is an hypothesis, which, if known, sanctions the conclusion l. For efficient search $\neg S$ defines a most general context in which C holds. ATMSs have found widespread use in AI applications such as diagnosis, qualitative physics, and constraint propagation.

The ATMS is purely propositional and does not itself perform any non-monotonic inference — that is left to rest of the implementation. One of the most important approaches to non-monotonicity is circumscription [18]. We have developed a Minimality Maintenance System which has the same functionality of the ATMS except with respect to some circumscriptive policy. An MMS is provided a sequence of propositional clauses and amendments to the circumscriptive policy (i.e., which literals are minimized, fixed, or allowed to vary) interspersed with queries. Given a query l, the MMS is expected to respond with every shortest clause S for which $S \vee l$ is a logical consequence of circumscribing the clause sequence with respect to the current circumscriptive policy. The advantages of this are twofold: (1) the problem solver can now work with theories which are inherently circumscriptive, (2) (non-monotonic) functionality which used to have to be implemented in the inference engine can now shifted to the MMS. For example, diagnostic models which are inherently circumscriptive (such as [23]) can now be implemented directly. Also dynamic constraint satisfaction problems (DCSP) (such as [21]) can now be implemented directly. In addition, both of these systems restrict the form of the models and constraints to facilitate a special-purpose implementation. This restriction is removed when using the MMS.

Ginsberg [13] developed a circumscriptive theorem prover which, given a query Q and a theory T, determines whether Q follows from the circumscribed theory T. Ginsberg used an ATMS-like mechanism to support this task, however, he addresses a quite different task than we do. The MMS is different in three fundamental ways: (1) it performs the abduction task — for each query it returns a set of clauses which, if true, make the query true in the circumscribed theory, (2) it compiles its results so that answers to queries are immediate and

involve no additional reasoning — this makes queries very inexpensive, and (3) it is incremental.

We consider a propositional language $\mathcal{L}$ with a finite set of propositional symbols $\mathcal{S}$. The connectives $\wedge$, $\rightarrow$, $\vee$ and $\neg$ are defined in the usual way, as are the formulas of the language. A propositional symbol is also called a *positive literal*. The negation of a propositional symbol is called a *negative literal*. A *clause* is a disjunction of literals (positive or negative) with no literal repeated. A clause containing only one literal is called a unit clause. Let C and C' be two clauses. $C - C'$ denotes a clause whose literals are those in the difference of C and C'. The empty clause is noted $\emptyset$. For $\mathcal{X} \subset \mathcal{S}$, $\mathcal{X}^-$ denotes the set $\{\neg x \mid x \in \mathcal{X}\}$. Let Δ be a set of models of our language and Ω be a set of propositional formulae. We say that $\Delta \models \Omega$ just in the case every model of Δ assigns true to every formula of Ω. For a set of propositional formulae Σ of $\mathcal{L}$, the set of models of the language $\mathcal{L}$ which assign true to every formula of Σ is denoted $[\Sigma]$. For the purpose of defining propositional circumscription, the set of propositional symbols $\mathcal{S}$ is partitioned into three distinguished subsets: the minimized set $\mathcal{P}$, the varying set $\mathcal{V}$, and the fixed set $\mathcal{F}$. A circumscription policy is defined using minimal models semantics as follows:

Definition 1 *Let M and N be two models of a set of propositional formulae Σ. M is smaller than N noted $M \leq N$ iff:*

1. N assigns true to every propositional symbol of $\mathcal{P}$ for which M assigns true.

2. M and N assign the same truth value to every propositional symbol of $\mathcal{F}$.

A model M of a propositional formulae Σ is minimal if every model smaller than M assigns the same truth value to every propositional symbol of $\mathcal{P}$. The set of minimal models of Σ is notated $CIRC(\Sigma)$.

In order to extend the functionality of ATMSs for circumscriptive theories it is necessary to restate the definition of a support used in ATMSs [26].

Definition 2 *Let $\mathcal{M}$ be a set of models of the propositional language $\mathcal{L}$. A clause C is a $\mathcal{M}$-support for a clause K if the two following conditions hold:*

1. $\mathcal{M} \models \neg C \rightarrow K$.

2. $\mathcal{M} \not\models C$.

A minimal $\mathcal{M}$-support for a clause K is a $\mathcal{M}$-support of K such that no proper subclause is a $\mathcal{M}$-support of K.

The definition of a $\mathcal{M}$-support generalizes the definition of a support used by Reiter and de Kleer [26]. For a set of propositional formulae Σ, a $[\Sigma]$-support is simply called a support and corresponds to Reiter's and de Kleer's definition of a support given Σ. One may consider $\neg C$ as a "simplest" conjecture which "explains" K given Σ. Explanations are conjunctions of literals. A simplest explanation is one for which no proper subconjunct is an explanation. It is important to observe that a minimal support clause C for a clause K is relative to Σ. If Σ incompletely characterizes a state of affairs, then $\neg C$ might not be the simplest conjecture which sanctions K in that state of affairs. Circumscribing Σ copes with incomplete knowledge by jumping to conclusions which do not follow from Σ alone. We can exploit circumscription policies to modify the set of explanations by considering $CIRC(\Sigma)$-supports, called here, circumscriptive-supports. A minimal circumscriptive-support C for a clause K conjectures that $\neg C$ is a simplest hypothesis which explains K given Σ and the circumscription policy. Introducing circumscription modifies the set of explanations in various ways. It may be the case that a clause C is a minimal support for a clause K although C is not a circumscriptive-support for K. Therefore, the circumscriptive theory eliminates $\neg C$ from the set of explanations for K. Another typical case is when a minimal support C for a clause K is still a circumscriptive-support for K but is not a minimal circumscriptive-support. In such a case the circumscriptive theory provides a shorter explanation for K than the conjunction of literals appearing in $\neg C$. It may also be the case that a clause C is a minimal circumscriptive-support for a clause K although C is not a support for K. Therefore the circumscriptive theory generates the new explanation $\neg C$ for K.

The following examples highlight some of the differences between minimal supports and minimal circumscriptive supports:

Example 1. Suppose that Σ_1 consists of:

$$\{[f_1 \vee p_1 \vee \neg p_2], [f_2 \vee p_1], [f_3 \vee f_1 \vee p_1], [p_2 \vee v_1]\}.$$

Let us search for explanations for the literals p_2, v_1 and $\neg p_1$. Considering $[\Sigma_1]$ leads us to the following minimal supports for them (we use MIN(x) to indicate the minimal supports for x):

MIN(p_2): $\neg p_2$, v_1.
MIN(v_1): p_2, $\neg v_1$.
MIN($\neg p_1$): p_1.

Consider the circumscriptive policy for Σ_1: $\mathcal{P} = \{p_1, p_2\}$, $\mathcal{V} = \{v_1\}$, $\mathcal{F} = \{f_1, f_2, f_3\}$. For simplicity we identify a model and the corresponding set of positive literals which are assigned to true: The minimal models of Σ_1 are: $\{v_1, f_1, f_2, f_3\}$, $\{v_1, f_1, f_2\}$, $M_3 : \{v_1, f_2, f_3\}$, $M_4 : \{p_1, v_1, f_2\}$, $M_5 : \{p_1, v_1, f_1, f_3\}$. Every minimal model of Σ_1 assigns true to $\neg p_2$ and true to v_1. $\neg p_1$ exactly holds in all minimal models of Σ_1 which either assigns true to the conjunction $f_1 \wedge f_2$ or to the conjunction $f_2 \wedge f_3$. In particular, $CIRC(\Sigma_1)$ sanctions the following conclusions which do not follow from Σ_1 alone: $\neg p_2$, v_1, $f_1 \wedge f_2 \rightarrow \neg p_1$, $f_2 \wedge f_3 \rightarrow \neg p_1$. Since $\neg p_2$ follows from the circumscription policy there is no minimal circumscriptive-supports for p_2. Since v_1 follows from the circumscription policy, the empty clause is the minimal circumscriptive-support for v_1. Although the circumscription policy does not conjecture that $\neg p_1$ holds, it restricts the cases where p_1 holds and provides non-trivial circumscriptive supports for $\neg p_1$ (we use MINC(x) to refer to the minimal circumscriptive supports for x):

MINC(p_2): none.
MINC(v_1): $\emptyset$.
MINC($\neg p_1$): p_1, $\neg f_1 \vee \neg f_2$, $\neg f_2 \vee \neg f_3$.
This example illustrates the three different ways the circumscription policy modifies the set of explanations. The circumscriptive theory (1) eliminates explanations for p_2 (2) shortens explanations for v_1 and (3) introduces new explanations for $\neg p_1$.

Example 2. Consider a simple circuit comprising two components: a lightbulb and a battery. A diagnosis engine has to conjecture the potential causes for failures. To achieve this task the diagnosis engine is given the following set Σ_3 of propositional formulae involving the ground literals $\{ab(battery), ab(bulb), light, current\}$:

$$\neg ab(battery) \rightarrow current,$$

$$current \wedge \neg ab(bulb) \rightarrow light,$$

$$light \wedge \neg ab(bulb) \rightarrow current,$$

$$light.$$

MIN($\neg current$): $current$.
MIN($\neg ab(bulb)$): $ab(bulb)$.
MIN($\neg ab(battery)$): $ab(battery)$.

A circumscriptive theory of diagnosis [23; 25] introduces a parsimony principle which allows us to jump to the conclusion that components are not abnormal when there is no evidence to the contrary. This leads us to minimize the cases where the literals $ab(bulb)$ and $ab(battery)$ hold:

$$\mathcal{P} = \{ab(bulb), ab(battery)\}.$$

There is a whole spectrum of parsimony principles depending on the choice of a particular circumscription policy (i.e., the choice of the varying and fixed sets). As the varying set increases and the fixed set decreases more components are exonerated (i.e., shown to be not abnormal). The parsimony principle introduced in [25] is at one end of the spectrum and corresponds to the choice where the fixed set is empty. [23] motivates the choice of a different parsimony principle corresponding to the circumscription policy for which the varying set is empty:

$$\mathcal{V} = \{\}, \quad \mathcal{F} = \{current, light\}.$$

$CIRC(\Sigma_3)$ circumscribes the abnormal behavior of the bulb and the battery to the case where there is no current allowing us to infer $ab(bulb) \rightarrow \neg current$ and $ab(battery) \rightarrow \neg current$. As a consequence the circumscribed theory sanctions the two following conclusions, called alibis (because the normality of one component exonerates another) [23], which exclude the manifestation of single faults: $\neg ab(bulb) \rightarrow \neg ab(battery)$ and $\neg ab(battery) \rightarrow \neg ab(bulb)$. Therefore:
MINC($\neg current$): $current$, $\neg ab(bulb)$, $\neg ab(battery)$.
MINC($\neg ab(bulb)$): $\neg current$, $ab(bulb)$, $ab(battery)$.
MINC($\neg ab(battery)$): $\neg current$, $ab(bulb)$, $ab(battery)$.

3 Characterizing circumscriptive supports

A characterization of minimal supports is given in [26] using the notion of prime implicates. Reiter and de Kleer show that there is a direct mapping between minimal supports for unit clauses and the set of prime implicates. We generalize this result to provide a useful characterization of minimal circumscriptive supports using prime implicates. In order to provide such a characterization we need some intermediate results.

3.1 Circumscriptive implicates

As an intermediate step we introduce the notion of prime $\mathcal{M}$-implicate which generalizes the notion of prime implicate.

Definition 3 *Let $\mathcal{M}$ be a set of models of the propositional language $\mathcal{L}$. A $\mathcal{M}$-implicate is a clause C such that: $\mathcal{M} \models C$. A prime $\mathcal{M}$-implicate is an $\mathcal{M}$-implicate such that no proper subclause is a $\mathcal{M}$-implicate. The set of prime $\mathcal{M}$-implicates is noted $PI(\mathcal{M})$.*

For a set of propositional formulae Σ, prime $[\Sigma]$-implicates correspond to the notion of prime-implicates used in [26]. We call $CIRC(\Sigma)$-implicates circumscriptive-implicates. Considering prime $\mathcal{M}$-implicates provides us with the following characterization of minimal $\mathcal{M}$-supports for unit clauses:

Theorem 1 *Let $\mathcal{M}$ be a set of models of $\mathcal{L}$. A clause C is a minimal $\mathcal{M}$-support for a unit clause $\{l\}$, iff there is a prime $\mathcal{M}$-implicate Π of $\mathcal{M}$ such as $C = \Pi - \{l\}$.*

Let Σ be a set of propositional formulae. Applying Theorem 1 to $CIRC(\Sigma)$ shows that there is a direct mapping between minimal circumscriptive-supports and prime circumscriptive-implicates. Therefore the characterization of prime circumscriptive-supports is reduced to the characterization of prime circumscriptive-implicates.

Example 1 (continued). Applying Theorem 1 to $CIRC(\Sigma_1)$ shows that the (non-trivial) minimal circumscriptive-supports can be deduced from the following set of (non-tautological) prime circumscriptive-implicates of $[\Sigma_1]$:
$\{\neg p_2, v_1, [\neg p_1 \vee \neg f_1 \vee \neg f_3], [\neg p_1 \vee \neg f_2 \vee \neg f_3], [f_2 \vee p_1], [f_1 \vee f_3 \vee p_1]\}$.

3.2 Minimizers

Our basic strategy is to augment the set of propositional formulae Σ with an additional set of formulae Σ' such that the prime circumscriptive implicates are the prime implicates of $\Sigma \cup \Sigma'$. Σ' will include one additional minimizer formula for each minimized symbol. Consider a set of propositional formulae Σ and a literal p in the minimized set $\mathcal{P}$. The role of the minimizer for p is to characterize when the circumscription policy conjectures $\neg p$. Intuitively the minimization process resulting from the circumscription policy leads us to conclude $\neg p$ when

there is no evidence to the contrary. Therefore, the minimizer for p is the 'weakest' DNF formula which ensures that there is no evidence that p holds. But the minimal support for p provides an easy way to construct the weakest such formula. More precisely, let C be a minimal support for p. Since $[\Sigma] \models \neg C \rightarrow p$, $\neg C$ is a minimal condition that must be met in order that Σ sanctions p. If all the minimal supports for p hold, then there is no longer any evidence that p holds. Under the condition that the conjunction of all minimal supports for p hold the circumscription policy sanctions $\neg p$. However, the circumscription policy lets some literals vary in the minimization. Evidence that p holds based on set of varying literals are disregarded. This motivates the following definition of minimizer:

Definition 4 *Let Σ be a set of propositional formulae, and p be a literal of $\mathcal{P}$. The minimizer of p, noted $\mu(p)$, is the conjunction of all the minimal support clauses of p with respect to Σ which contain only literals in $\mathcal{P} \cup \mathcal{F} \cup \mathcal{F}^-$.*

Theorem 2 *Let Σ be a set of propositional formulae.*

$$CIRC(\Sigma) = [\Sigma \cup \{\mu(p) \rightarrow \neg p \mid p \in \mathcal{P}\}].$$

Example 1 (continued). There are three prime-implicates containing the literal p_1: $\neg p_1 \vee p_1$, $f_2 \vee p_1$ and $f_1 \vee f_3 \vee p_1$. Applying Theorem 1 to $[\Sigma]$ shows that $\text{MIN}(p_1) = \{\neg p_1, f_2, f_1 \vee f_3\}$. From the definition of the minimizer of p_1 it follows that:

$$\mu(p_1) \equiv f_2 \wedge [f_1 \vee f_3].$$

From theorem 2:

$$CIRC(\Sigma_1) \models \neg p_1 \equiv f_2 \wedge [f_1 \vee f_3].$$

The following Corollary enables the construction of the MMS:

Corollary 1 *Let Σ be a set of propositional formulae. A clause C is a minimal circumscriptive support for a literal l iff there is prime implicate Π of $[\Sigma \cup \{\mu(p) \rightarrow \neg p \mid p \in \mathcal{P}\}]$, such as $C = \Pi - \{l\}$.*

4 The non-incremental algorithm

Our incremental MMS algorithm includes a number of not immediately obvious innovations to preserve as much previous work as possible. Therefore, we first describe a direct MMS algorithm which follows the theorems directly. The MMS is given a set of propositional formulae and asked to produce the minimal circumscriptive supports for unit clauses.

1. The formulae are reduced to their prime implicates, Π.
2. Find all clauses in Π containing at least one literal from $\mathcal{P}$ where the rest of the literals are from $\mathcal{P} \cup \mathcal{F} \cup \mathcal{F}^-$. From these clauses construct the minimizers for each p.
3. Construct the formula $\mu(p) \rightarrow \neg p$ for each $p \in \mathcal{P}$ having a minimizer.
4. Compute the prime implicates of Π and the formulae of the previous step.

The circumscriptive prime implicates can now be read of directly from the final set of prime implicates.

5 An incremental algorithm

Our incremental algorithm uses the incremental prime implicate algorithm described in [9]. The fact that the addition of any formula always reduces the number of models, makes incremental ATMS algorithms easy to design. However, the direct incremental analog does not work for a MMS because the addition of a new formula can result in more models of the circumscribed theory. Fortunately, with the use of prime implicates we can pinpoint which minimized symbols are affected and therefore preserve as much as possible of the previous MMS computations.

In our MMS implementation all symbols are presumed to vary unless indicated otherwise. We create a new symbol $A(p)$ for every minimized symbol p. The idea is that every minimizer we construct for p will be made contingent on $\neg A(p)$ such that it can easily be changed as new information is obtained about p. We designate the set of $A(p)$'s as $\mathcal{A}$. The MMS maintains a single data base Π of prime implicates.

Our MMS supports the following transactions:

1. Request for the minimal circumscriptive supports for a literal.
2. Changing a symbol between minimized, varying and fixed.
3. Adding new formulae.

To find the minimal circumscriptive supports for a literal l we perform the following steps:

1. Construct Q consisting of all clauses in Π which mention l.
2. Remove all elements of $\mathcal{A} \cup \{l\}$ from the clauses in Q.
3. Remove all clauses from Q which are subsumed by some other.

Changing a symbol p from minimized to varying is achieved by removing all clauses from Π containing $A(p)$ and removing $A(p)$ from $\mathcal{A}$. Changing a symbol p from varying to minimized is achieved by creating the minimizer $\mu(p)$ and symbol $A(p)$ and updating Π by computing the prime implicates of Π and $\neg A(p) \rightarrow [\mu(p) \rightarrow \neg p]$. The other cases follow similarly and are described in the long version of the paper.

The following is a sketch of the algorithm to add new formulae to an existing data base of prime implicates Π. A key to understanding the algorithm is to observe that the MMS algorithm maintains the invariant that the minimizer of every $A(p)$ is always empty.

1. The new formulae are reduced to their prime implicates π.

2. Using the incremental prime implicate algorithm replace Π with the prime implicates of $\Pi \cup \pi$. If Π does not change, then we are done.
3. Construct the minimizer for every $A(p) \in \mathcal{A}$. Find all clauses in Π containing at least one literal from $\mathcal{A}$ where the rest of the literals are from $\mathcal{P} \cup \mathcal{A} \cup \mathcal{F} \cup \mathcal{F}^-$. All such clauses must be new clauses resulting from the addition of π. If there are none, then we are done. From these clauses construct the minimizers for the $A(p)$.
4. For each $A(p)$ having a non-empty minimizer, construct a new symbol $A'(p)$ and the formula $\neg A'(p) \rightarrow [\mu(A(p)) \rightarrow \neg A(p)]$.
5. Using the incremental prime implicate algorithm replace Π with the prime implicates of Π and the new formulae produced in the preceding step.
6. Remove each minimized $A(p)$ from $\mathcal{A}$ and remove from Π all clauses mentioning it. Then rename each $A'(p)$ to $A(p)$.

Step 3 incorporates the minimizer formulation introduced in this paper. The surrounding steps focus on preserving previous work. Thus if a new formula is added which is propositionally entailed, then the algorithm terminates at Step 2. In Step 3, if the new formulae do not affect the minimizers, then the algorithm terminates. Step 4 only computes the changes to the minimizers induced by the new formulae. Thus, symbols which are unaffected by the new formulae produce no additional work, and Step 4 constructs only the incremental change to the minimizers of the affected symbols. Therefore, we see that circumscription which is often viewed as a global operation on a theory can, in fact, be reduced to a sequence of local operations.

Let us do example 1. The circumscriptive policy is: $\mathcal{P} = \{p_1, p_2\}, \mathcal{V} = \{v_1\}, \mathcal{F} = \{f_1, f_2, f_3\}$.

Initially there are no clauses. The minimizer for both p_1 and p_2 is thus empty and thus Π contains (we will only include the non-tautologous prime implicates):

$$\{A(p_1) \vee \neg p_1, A(p_2) \vee \neg p_2\}.$$

Thus, $\neg p_1$ and $\neg p_2$ hold in the circumscribed theory. Now the MMS receives the clause: $f_1 \vee p_1 \vee \neg p_2$. This clause is itself a prime implicate. The other resulting prime implicate, $A(p_1) \vee \neg p_2 \vee f_1$, does not contribute to the minimizer of $A(p_1)$, therefore no additional minimizer is needed. Now the MMS receives the clause: $f_2 \vee p_1$. This clause is itself a prime implicate. The other prime implicate $A(p_1) \vee f_2$ will result in a minimizer for $A(p_1)$. Adding the minimizer results in the following two prime implicates: $\neg p_1 \vee \neg f_2 \vee A'(p_1)$ and $\neg p_2 \vee f_1 \vee \neg A'(p_1)$. Now we see that $\neg p_1$ and $\neg p_2$ no longer hold universally but only under certain conditions. For example, $\neg p_1$ holds only when f_2 is true which exactly what we would intuitively expect given that the clause $f_2 \vee p_1$ has been added. For expository purposes we will not rename $A'(p_1)$ to $A(p_1)$. Now the MMS is provided the clause $f_3 \vee f_1 \vee p_1$. But this clause resolves with no other so it itself becomes a prime implicate. However, this clause does affect p_1. We have to construct another symbol $A''(p_1)$ and adding the minimizer for $A'(p_1)$ results in the following three new prime implicates: $\neg p_1 \vee \neg f_1 \vee \neg f_2 \vee A''(p_1)$, $\neg p_1 \vee \neg f_2 \vee \neg f_3 \vee A''(p_1)$, $\neg p_2 \vee f_1 \vee \neg f_2 \vee A''(p_1)$. Notice that for the first time $\neg p_2$ is no longer holds universally. Finally, the MMS is provided the clause $p_2 \vee v_1$. In addition to itself, this results in new prime implicates: $f_1 \vee \neg f_2 \vee \neg f_3 \vee v_1 \vee A''(p_1)$, $A(p_2) \vee v_1$, $p_1 \vee f_1 \vee v_1$. However, none of these clauses affects the minimization. So no new minimizers need be constructed. A quick check will show that the resulting set of prime implicates is equivalent to those given earlier.

6 Conclusions

This paper has described the MMS, a general, efficient and incremental system for performing abduction on circumscribed theories. It performs exactly the right kind of analysis required for diagnosis and dynamic constraint satisfaction tasks. In addition, by being incremental, it fully supports the sequential character of these tasks — allowing new observations and constraints to be added while maximally preserving old work. This is achieved by recognizing that circumscription, which is usually thought of as a global operation, can be reduced to a set of (often) local minimizations and thus only those minimizers affected by a new observation or constraint provoke additional computation.

References

[1] Chang, C. and Lee R.C., *Symbolic Logic and Mechanical Theorem Proving*, (Academic Press, New York, 1973).

[2] de Kleer, J., An assumption-based truth maintenance system, *Artificial Intelligence* **28** (1986) 127–162. Also in *Readings in NonMonotonic Reasoning*, edited by Matthew L. Ginsberg, (Morgan Kaufman, 1987), 280–297.

[3] de Kleer, J., Extending the ATMS, *Artificial Intelligence* **28** (1986) 163–196.

[4] de Kleer, J., A practical clause management system, SSL Paper P88-00140, Xerox PARC.

[5] de Kleer, J., Forbus, K., and McAllester D., Tutorial notes on truth maintenance systems, IJCAI-89, Detroit, MI, 1989.

[6] de Kleer, J., A comparison of ATMS and CSP techniques, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI (August 1989).

[7] de Kleer, J. and K. Konolige, Eliminating the fixed predicates from a circumscription, *Artificial Intelligence* **39** (1989) 391–398.

[8] de Kleer, J., A Clause Management System based on Boolean Constraint Propagation and clause synthesis, 1990.

[9] de Kleer, J., A new incremental algorithm for generating prime implicates, *Proceedings AAAI-92*, San Jose, CA (1992).

[10] Doyle, J., A truth maintenance system, *Artificial Intelligence* **12** (1979) 231–272.

[11] Dressler, O., An Extended Basic ATMS, *Proceedings of the second international workshop on Non-Monotonic Reasoning*, Springer, LNCS 346, 143–163, (1988).

[12] Gelfond, M., Lifschitz,V., Compiling circumscriptive theories into logic programs, *Proceedings of the second international workshop on Non-Monotonic Reasoning*, Springer, LNCS 346, 143–163, (1988).

[13] Ginsberg, M.L., A circumscriptive theorem prover, *Proceedings of the second international workshop on Non-Monotonic Reasoning*, Springer, LNCS 346, 100–114, (1988).

[14] Kean, A. and Tsiknis, G., An incremental method for generating prime implicants/implicates, University of British Columbia Technical Report, TR-88-16, 1988.

[15] Kohavi, Z., *Switching and Finite Automata Theory* (McGraw-Hill, 1978).

[16] Lifschitz, V., Computing circumscription, *Proceedings IJCAI-85*, Los Angeles, CA (1985) 121–127.

[17] Lifschitz, V., Pointwise circumscription, *Proceedings AAAI-86*, Philadelphia, PA (1986) 406–410.

[18] McCarthy,J., Circumscription-A form of non-monotonic reasoning, *Artificial Intelligence* **13** (1985) 27–39.

[19] McCarthy, J., Applications of circumscription to formalizing common-sense knowledge, *Artificial Intelligence* **28** (1986) 89–116.

[20] Mackworth, A.K., Constraint satisfaction, *Encyclopedia of Artificial Intelligence*, edited by S.C. Shapiro, (John Wiley and Son, 1987) 205-211.

[21] Mittal, S. and Falkenhainer, B., Dynamic constraint satisfaction problems, *Proceedings AAAI-90*, Boston, (July 29-August 3, 1990).

[22] Przymusinski, T.C., An algorithm to compute circumscription, *Artificial Intelligence* **38** (1989) 49–73.

[23] Raiman, O., A circumscribed diagnosis engine, in: *Expert Systems in Engineering* , Lecture Notes in Artificial Intelligence, G. Gottlob, W. Nedjl (Eds), Springer-Verlag.

[24] Reiter, R., A logic for default reasoning, *Artificial Intelligence* **13**(1980) 81–132.

[25] Reiter, R., A theory of diagnosis from first principles, *Artificial Intelligence* **32**(1987) 57–95.

[26] Reiter, R. and de Kleer, J., Foundations of assumption-based truth maintenance systems: Preliminary report, in: *Proceedings AAAI-87*, Seattle, WA (July, 1987), 183–188.

[27] Tison, P., Generalized consensus theory and application to the minimization of boolean functions, *IEEE transactions on electronic computers* **4** (August 1967) 446-456.

7 Appendix: Proofs of Theorems

Theorem 1 *Let $\mathcal{M}$ be a set of models of $\mathcal{L}$. A clause C is a minimal $\mathcal{M}$-support for a unit clause $\{l\}$, iff there is a prime $\mathcal{M}$-implicate Π of $\mathcal{M}$ such as $C = \Pi - \{l\}$.*

Proof. $\Rightarrow$ Let C be a minimal $\mathcal{M}$-support of the unit clause $\{l\}$. Consider the clause $\Pi = C \cup \{l\}$. Since C is a $\mathcal{M}$-support for $\{l\}$, Π is a $\mathcal{M}$-implicate. Let us show that Π is a prime $\mathcal{M}$-implicate. Assume there is a proper subclause K of Π which is also a $\mathcal{M}$-implicate. If K is a subclause of C then C cannot be a support of $\{l\}$. If $K = C' \cup \{l\}$ where C' is a proper subclause of C, then C' is a $\mathcal{M}$-support of $\{l\}$. Therefore C is not a minimal $\mathcal{M}$-support of $\{l\}$.
$\Leftarrow$ Let $\Pi = C \cup \{l\}$ be a prime $\mathcal{M}$-implicate. Let us show that C is minimal $\mathcal{M}$-support of $\{l\}$. Since no proper subclause of Π is a $\mathcal{M}$-implicate, C is a $\mathcal{M}$-support of $\{l\}$. Moreover if a proper subclause of C is a $\mathcal{M}$-support of $\{l\}$, then there is a proper subclause of Π which is a $\mathcal{M}$-implicate.

Lemma 1 *Let Σ be a set of propositional formulae, and p be a literal of $\mathcal{P}$.*

$$CIRC(\Sigma) \models \neg p \equiv \mu(p).$$

Proof. $\Rightarrow$ Let M be a model of Σ. If M assigns true to $\neg p$ then M must assign true to all the minimal supports of p. In particular M assigns true to $\mu(p)$. Thus, every model of Σ and in particular every minimal model of Σ assigns true to $\neg p \rightarrow \mu(p)$.
$\Leftarrow$ Let M be a minimal model of Σ. Consider the case where M assigns true to p. Let us show that M assigns false to $\mu(p)$. Let C be the clause comprising exactly the set of literals in $\mathcal{P} \cup \mathcal{F} \cup \mathcal{F}^-$ for which M assigns false. M also assigns false to C. Since M is a model of Σ:

$$[\Sigma] \not\models C.$$

Since M is a minimal model of Σ, every model of Σ which assigns false to C must assign true to the propositional symbol p:

$$[\Sigma] \models C \vee p.$$

Therefore the clause C is a support for the positive literal p. Moreover there exist a subclause C' of C which is a minimal support of p. Since M assigns false to the clause C a fortiori M also assigns false to C'. C' like C contains only literals in $\mathcal{P} \cup \mathcal{F} \cup \mathcal{F}^-$. By definition of the minimizer of p, $\mu(p) \rightarrow C'$. Thus, M must assign false to $\mu(p)$ and therefore M assigns true to $\mu(p) \rightarrow \neg p$.

Theorem 2 *Let Σ be a set of propositional formulae.*

$$CIRC(\Sigma) = [\Sigma \cup \{\mu(p) \rightarrow \neg p \mid p \in \mathcal{P}\}].$$

Proof. Let us show that the minimal models of Σ exactly correspond to the models of $\Sigma \cup \{\mu(p) \rightarrow \neg p \mid p \in \mathcal{P}\}$.

$\Rightarrow$ Let M be a minimal model of Σ. Lemma 1 shows that M is a model of $\{\mu(p) \rightarrow \neg p \mid p \in \mathcal{P}\}$. Moreover M is also a model of Σ. Thus every minimal model of Σ is a model of $\Sigma \cup \{\mu(p) \rightarrow \neg p \mid p \in \mathcal{P}\}$

$\Leftarrow$ Let M be a model of $\Sigma \cup \{\mathcal{M}(p) \rightarrow \neg p | p \in \mathcal{P}\}$. M is a model of Σ. Let p be a propositional symbol of $\mathcal{P}$ for which M assigns true. M must assign false to $\mu(p)$. By the definition of the minimizer of p, there exist a minimal support C of p which contains only literals in $P \cup \mathcal{F} \cup \mathcal{F}^-$, such as M assigns false to C. Let N be a model of Σ smaller than M. N assigns the same truth values as M to the literals of $\mathcal{F} \cup \mathcal{F}^-$, and N assigns false to all the positive literals of $\mathcal{P}$ for which M assigns false. Therefore N like M assigns false to C. Since C is a support for p, N will assign true to p. Thus every model of Σ smaller than M assigns true to p. Therefore every model of Σ smaller than N assigns the same truth value to every propositional symbol of $\mathcal{P}$. M is a minimal model of Σ.

Search through Systematic Set Enumeration

Ron Rymon*
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Abstract

In many problem domains, solutions take the form of *unordered* sets. We present the *Set-Enumeration* (SE)-tree – a vehicle for representing sets and/or enumerating them in a best-first fashion. We demonstrate its usefulness as the basis for a unifying search-based framework for domains where minimal (maximal) elements of a power set are targeted, where minimal (maximal) partial instantiations of a set of variables are sought, or where a composite decision is not dependent on the *order* in which its primitive component-decisions are taken. Particular instantiations of SE-tree-based algorithms for some AI problem domains are used to demonstrate the general features of the approach. These algorithms are compared theoretically and empirically with current algorithms.

1 INTRODUCTION

Many computer science problems admit solutions which are elements of a given power-set. Typically, such sets are required to satisfy some problem-specific criterion which designates them as solutions. In many cases, such criteria either include, or are augmented with, some minimality/maximality requirement. Consider, for example, the Hitting-Set (HS) problem [Karp 72]. Given a collection of sets, solutions are required to have a non-empty intersection with each member of the collection. In applications of the HS problem, *interesting* solutions are typically minimal with respect to set inclusion. In a more general class of problems, solutions are partial instantiations of a set of variables. A hitting-set, for example, can also be described as a membership-based mapping from the underlying set of primitive elements to $\{0, 1\}$. More generally, variables can be instantiated from an arbitrary domain.

Researchers in Artificial Intelligence (AI) have also made use of such abstract problems in their models. The HS problem, for example, was used by [Reiter 87] in his formalization of diagnosis. In a newer characterization, diagnoses are viewed as partial assignments of state to components [de Kleer et al. 90]. Many other AI problems are, or could be, formulated so as to admit sets as solutions.

Our goal in introducing the *Set-Enumeration* (SE)-tree is to provide a unified search-based framework for solving such problems, albeit their problem-independent solution criteria. SE-tree-based algorithms for different problems will share their skeletal structure, but will each use additional domain-specific *tactics*. Furthermore, at a certain level of abstraction, even those tactics are general and can be shared across domains. General tactics identified here include pruning rules which exploit the SE-tree structure, exploration policies, and problem decomposition methods. Incremental versions of SE-tree-based algorithms can be constructed for some problem domains. In what follows, we use particular instantiations of SE-tree-based algorithms to demonstrate the general features of the approach.

Consistency-based formulations of diagnosis will serve as a working example for most of this paper. Section 2 begins with a formal description of the basic SE-tree. Section 3 presents an SE-tree-based hitting-sets algorithm (SE-HS) for Reiter's original formulation. Being best-first, it can use any of a number of exploration policies. This algorithm is first contrasted, as is, with the original algorithm. The SE-tree structure is then used to improve SE-HS by pruning away unpromising parts of the search space. We conclude with an empirical comparison of the two algorithms.

In Section 4, we extend the SE-tree to fit the more general class of problems, where solutions take the form of partially instantiated sets of variables. Section 5 presents an extended version of SE-HS for a newer

*Address for correspondence: Ron Rymon, Computer and Information Science, Room 423C, 3401 Walnut Street, Philadelphia PA 19104, e-mail: rymon@linc.cis.upenn.edu.

characterization of diagnoses [de Kleer et al. 90]. Although derived from a very general search framework, this algorithm corresponds to a prime implicate generation algorithm proposed by [Slagle et al. 70], and is empirically shown to perform quite well compared to a recent algorithm [Ngair 92]. Unlike Slagle et al.'s algorithm, the extended SE-HS can work under diagnostic theories with *multiple* fault modes, can use a variety of exploration policies for focusing purposes, and has an incremental version. Furthermore, we subsequently augment it with a problem decomposition tactic, thereby obtaining an improved version of Slagle et al.'s algorithm. Finally, we briefly review potential use of the SE-tree in abductive diagnostic frameworks.

In Section 6, we contrast features of the SE-tree with decision trees in the context of learning classification rules from examples. For lack of space, the scope of this study is very limited and the reader is referred to [Rymon 92b] for a more detailed analysis and empirical evaluation.

2 THE BASIC SE-TREE

The *Set-Enumeration* (SE)-tree is a vehicle for representing and/or enumerating sets in a best-first fashion. The *complete* SE-tree systematically enumerates elements of a power-set using a pre-imposed order on the underlying set of elements. In problems where the search space is a subset of that power-set that is (or can be) closed under set-inclusion, the SE-tree induces a complete irredundant search technique. Let E be the underlying set of elements. We first index E's elements using a one-to-one function $ind : E \rightarrow \mathbb{N}$. Then, given any subset $S \subseteq E$, we define its SE-tree view:

Definition 2.1 A Node's View

$$View(ind,S) \stackrel{\text{def}}{=} \{ e \in E \mid ind(e) > \max_{e' \in S} ind(e') \}$$

Definition 2.2 A Basic Set Enumeration Tree

Let F be a collection of sets that is closed under $\subseteq$ (i.e. for every $S \in F$, if $S' \subseteq S$ then $S' \in F$). T is a Set Enumeration tree for F iff:

1. *The root of T is labeled by the empty set;*
2. *The children of a node labeled S in T are* $\{ S \cup \{e\} \in F \mid e \in View(ind,S) \}$.

Figure 1 illustrates an SE-tree for the complete power-set of $\{1,2,3,4\}$. Note that restricting a node's expansion to its *View*, ensures that every set is *uniquely* explored within the tree. By itself, the idea of using an imposed order is not new; it is used for similar purposes in many specific algorithms. Our contribution is in identifying the SE-tree as a recurring search structure, thereby facilitating its use in a general framework and the sharing of particular tactics.

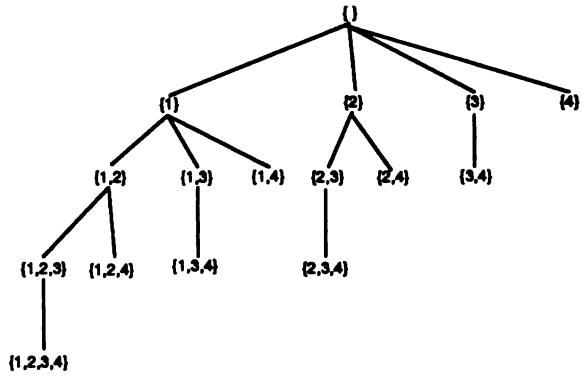


Figure 1: SE-tree for $\mathcal{P}(\{1,2,3,4\})$

Notice also that the SE-tree can be used as a data structure for *caching* unordered sets, and as an effective means of checking whether a new set is subsumed by any of those already cached. [de Kleer 92] has made such use of an SE-tree and reports significant improvements in run-time. As a caching device, the SE-tree is a special case of Knuth's *trie* data structure [Knuth 73], originally offered for *ordered* sets. While we too use the SE-tree for caching solutions and for subsumption checking, our main objective in this paper is its use in a search framework.

3 AN SE-TREE-BASED HITTING-SET ALGORITHM

In this section, we demonstrate the use of the basic SE-tree structure for a hitting-set algorithm in the context of Reiter's theory of diagnosis. We open with a brief introduction of Reiter's theory, to the point in which a hitting-set problem is formulated. An SE-tree-based algorithm (SE-HS) is then contrasted with the dag-based algorithm proposed by [Reiter 87, Greiner et al. 89] to show that a large number of calls to a subsumption checking procedure can be saved. Then, the SE-tree systematicity allows improving SE-HS via a domain-specific pruning rule. Empirical comparison of the improved SE-HS with the dag-based implementation of [Greiner et al. 89] supports our claims.

3.1 REITER'S THEORY OF DIAGNOSIS

Reiter's theory of diagnosis [Reiter 87] is among the the most widely referenced logic-based approaches to model-based diagnosis. For lack of space, we shall only present the concepts and theorem which Reiter uses to derive his hitting-set algorithm.

Definition 3.1 A Diagnostic Problem [Reiter 87]

A diagnostic problem is a triple (SD,COMPS,OBS)*:*

1. SD – *the system description, is a set of first order sentences;*
2. COMPS $\stackrel{\text{def}}{=} \{c_i\}_{i=1}^{n}$ – *the system's components, is a finite set of constants; and*
3. OBS – *the observations, is also a set of first order sentences.*

The language in which diagnostic problems are expressed is thus first order, and is augmented with an extra AB predicate (for abnormal).

Definition 3.2 Conflict Set

Given a diagnostic problem, a conflict is a set of components that cannot all *be functioning correctly. Let* CONFLICTS *denote the collection of conflict sets.*

Theorem 3.3 [Reiter 87] *Given a diagnostic problem, minimal diagnoses are precisely the minimal hitting sets for* CONFLICTS.

Reiter's algorithm is an implementation of Theorem 3.3. In two steps, it first discovers conflicts, and then runs an HS algorithm on the conflicts discovered. We shall concentrate on the latter phase.

3.2 DAG-BASED APPROACH

Given a collection of conflict sets, Reiter's algorithm grows an HS-tree in which nodes represent partial hitting sets and leaves represent complete ones. To avoid highly redundant exploration, Reiter augments this basic algorithm with a set of rules for *reusing* and *pruning* nodes. [Greiner et al. 89] present a correction to this algorithm which uses a directed acyclic graph (dag). It proceeds as follows:

1. Let D represent a growing HS-dag. Label its root with an arbitrary $C \in$ CONFLICTS;
2. Process nodes in D in a breadth-first order. To process a node n:
   (a) Let $H(n)$ be the set of edge labels on the path from the root to n. If $H(n)$ hits all sets in CONFLICTS, mark it as a minimal hitting set. Otherwise, label n with the first set of CONFLICTS which is not hit by $H(n)$.
   (b) If n is labeled by Σ, generate a downward arc labeled by any $\sigma \in \Sigma$.

This algorithm is augmented with three types of rules for expanding a node n:

1. *Reusing:* If there is another node m for which $H(m) = H(n) \cup \{\sigma\}$, do not expand n, but rather link it to m, labeling that link with σ.
2. *Closing:* If there is a node m which is marked as a hitting set, such that $H(m) \subseteq H(n)$, then close n, i.e. do not expand it at all.
3. *Pruning:* If a set Σ is to label a node n and it has not been used previously, then try to prune D:
   (a) If there is a node m which has been labeled with a set S' such that $\Sigma \subseteq S'$, then relabel m with Σ. Prune all edges from m with arcs labeled with σs from $S'-\Sigma$.
   (b) Interchange S' and Σ in CONFLICTS.

3.3 SE-TREE-BASED ALTERNATIVE

SE-HS (Algorithm 3.4) is an SE-tree-based hitting set algorithm. In a best-first fashion, it explores nodes in an order conforming to some predetermined priority function. For that purpose, nodes along the tree's expanding fringe are kept in a priority queue and the next node to be expanded is accessed via the *Next-Best* operation. Prioritization allows implementation of various exploration policies, to be discussed shortly. Let us first assume that nodes are explored by their cardinality; i.e. breadth-first.

Algorithm 3.4 Finding Minimal Hitting Sets

Program SE-HS (CONFLICTS)

1. *Let HS* $\leftarrow \{\}$; *OPEN-NODES* $\leftarrow \{\{\}\}$
2. *Until OPEN-NODES is empty do*
3. *Expand (Next-Best(OPEN-NODES))*

Procedure Expand(S)

1. *Let Window(S)* $\leftarrow \{ c \mid c \in$ *View(ind,S)* $\}$
2. *For each* $c \in$ *Window(S) which is a member of some set from NYH(S) do*
3. *Unless there is* $S' \in HS$ *such that* $S' \subseteq S \cup \{c\}$
4. *If* $S \cup \{c\}$ *is a hitting set, add it to HS;*
5. *Otherwise, add it to OPEN-NODES.*

The main SE-HS program simply implements a best-first search. The algorithm's functionality is embodied in its *Expand* procedure, where the SE-tree structure is used, and where hitting sets are identified. In choosing viable expansions for a node labeled S, we restrict ourselves to components within S's *View*. Such components are also required to participate in conflicts not yet hit by S (denoted $NYH(S)$). Step 3 in *Expand* prunes away nodes subsumed by minimal hitting sets. It corresponds to the *closing* step in [Greiner et al. 89]. However, SE-HS *avoids* the redundancy for which *reusing* rules were devised, and does not require *pruning*.

Theorem 3.5 *If nodes are prioritized by their label's cardinality, then SE-HS is correct (produces all and only minimal hitting sets.)*

3.4 PROJECTED GAIN

The HS-dag algorithm uses three pruning rules, each of which is computationally expensive and requires numerous calls to a subsumption checking procedure. In examining the purpose of these rules, we note that (1) *Reusing* is aimed at avoiding redundancy in search, i.e. the phenomena that same part of the search space is repeatedly explored within the HS-tree. It requires comparing each new node to *every* previous node; (2) *Closing* is aimed at shutting nodes which are supersets of minimal diagnoses. For that purpose, if the HS-dag is explored breadth-first, each node will only have to be compared against previous minimal hitting sets; finally (3) *Pruning* is aimed at "correcting" the HS-dag from the effects of non-minimal conflict sets. The same effect could also be achieved *a priori*, by "sorting" CONFLICTS by cardinality.

As previously explained, while closing cannot be avoided, SE-HS requires neither reusing, nor pruning. Avoiding numerous calls to a subsumption checking procedure results in a tremendous improvement in run time (see Section 3.7).

3.5 EXPLORATION POLICIES

Due to its potentially exponential size, it may often be impossible to completely explore the space of sets. In such cases, it may be beneficial to characterize *partial* outputs of an SE-tree-based algorithm, given a variety of exploration policies

Definition 3.6 Correct Exploration Policy

An exploration policy is a priority function ψ, defined for each set. It is correct if whenever open nodes are so prioritized, the resulting algorithm is correct.

For the particular case of SE-HS, a variety of exploration policies are sensible.

Proposition 3.7 *Any monotonic function ψ (i.e. such that for every $S \subseteq S'$ we have $\psi(S) \leq \psi(S')$) is a correct exploration policy for SE-HS.*

We have already seen that exploration by cardinality is correct. Simpler diagnoses are explored first using this exploration policy. Other interesting policies include exploration by *probability* ($\psi(S) \stackrel{\text{def}}{=} Prob(S \text{ is a diagnosis})$), and by *utility* or some other monotonic external criterion imposed on sets.

3.6 PRUNING UNPROMISING PARTS OF THE SEARCH SPACE

So far, nodes were pruned only if subsumed by known hitting-sets, thereby using the minimality requirement and the monotonicity of the SE-tree with respect to set-inclusion. We have not used the systematic ordering of nodes in the SE-tree for that purpose. That ordering provides a *restriction* on node labels which can occur in a given node's sub-tree. More specifically, let S be a node's label, then the sub-tree rooted at that node will only have nodes whose labels are expansions of S with components from $View(ind, S)$. Thus, in choosing viable expansions for S, we can restrict ourselves to expansions such that every set that will *not* be hit by the expanded set will still contain components within its $View$ (and thus stand the chance of being hit by any of that node's descendants).

This is, in fact, a general feature of an SE-tree-based search program: the systematic enumeration embedded in the SE-tree structure allows us to ignore parts of the space which do not have the *potential* to lead to a solution.

To incorporate this pruning rule into SE-HS, it is sufficient to modify the node expansion routine.

Algorithm 3.8 Node Expansion (version 2)

Procedure Expand(S)

1. *Let* $Window(S) \leftarrow$
   $\{ c \mid c \in View(ind,S) \} \cap$
   $\{ c \mid ind(c) \leq \min_{S' \in NYH(S)} \max_{c' \in S'} ind(c') \}$
2. *For each* $c \in Window(S)$ *which is a member of some set from NYH(S) do*
3. *Unless there is* $S' \in HS$ *such that* $S' \subseteq S \cup \{c\}$
4. *If* $S \cup \{c\}$ *is a hitting set, add it to HS;*
5. *Otherwise, add it to OPEN-NODES.*

This algorithm is identical to Algorithm 3.4, except for the additional restriction in line 1. This change is an example of a domain-specific SE-tree-based pruning rule. The algorithm remains correct, but fewer nodes need be explored. We demonstrate this in the next section by way of an example, and via empirical experiments.

3.7 DEMONSTRATED GAIN

To demonstrate the advantages of SE-HS over the dag-based algorithm, we will first work through a complete example (taken from [Reiter 87]), and will then present the results of extensive empirical experiments.

Example 3.9 Consider the following collection of conflicts $\{\{2,4,5\}, \{1,2,3\}, \{1,3,5\}, \{2,4,6\}, \{2,4\}, \{2,3,5\}, \{1,6\}\}$ [Reiter 87]. Figure 2 depicts the corresponding HS-dag, where O's mark hitting sets, and X's denote closed nodes. The rightmost branch from the root was pruned by the last node to be explored (itself a descendant of that branch).

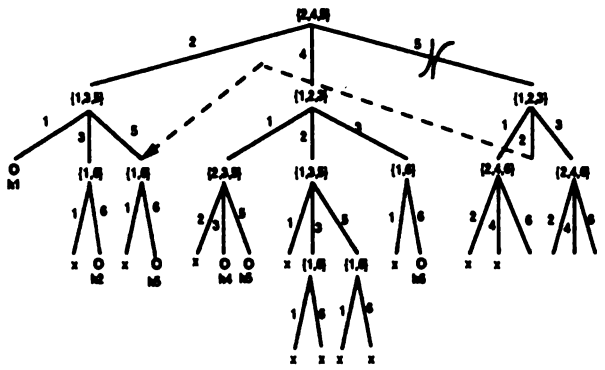


Figure 2: An HS-dag

In contrast, figure 3 shows an SE-tree for the same problem. In comparing, note the difference in notation: nodes in the HS-dag are labeled with the set that is chosen to be hit next whereas the SE-tree's labels are partial hitting sets. In the HS-dag the partial hitting sets label the path from the root to the particular node. The new pruning rule results in fewer nodes being explored: 16 in SE-HS versus 34 in HS-dag.

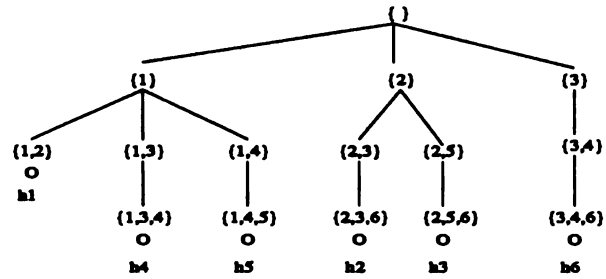


Figure 3: An SE-tree

In addition, we have empirically compared SE-HS's performance with that of an HS-dag implementation which was provided to us by Barbara Smith [Greiner et al. 89]. The run-time and number of nodes generated by each of the two implementations were tested on hundreds of randomly generated test cases. Test cases were generated using three parameters: number of conflicts (denoted #conf), number of components in each conflict (#lit), and overall number of components (#comp). Figures 4 and 5 use logarithmic scale to present one-way sensitivity analyses with respect to each of the parameters and with respect to both run-time (CPU seconds) and nodes generated. SE-HS's performance is indicated by the shadowed squares, and that of HS-dag by the open ones. Each data point was obtained by averaging each algorithm's performance on 10 random cases with same parameters.

4 THE EXTENDED SE-TREE

Sometimes the space being searched consists not of sets of components, but rather of sets of partially instantiated attributes (variables). We next extend the SE-tree accordingly.

Definition 4.1 Partial Descriptions

Let $ATTRS \stackrel{\text{def}}{=} \{A_i\}_{i=1}^{n}$ *be a set of attributes, with domains* $\{Dom(A_i)\}_{i=1}^{n}$. *A partial description is a subset of ATTRS, each of which is instantiated with one value from its domain. It is complete if all attributes are instantiated.*

Consider, for example, the space defined by 3 boolean attributes. The set $\{A_2{=}T\}$ is a partial description in that space. $\{A_1{=}T, A_2{=}F, A_3{=}F\}$ is a complete description.

As with its basic counterpart, to define the *extended* SE-tree we first impose an ordering (*ind*) on ATTRS, and define a node's *View* as all attributes ranked higher than the highest ranked attribute participating in that node. Then,

Definition 4.2 An Extended Set Enumeration Tree

Let F be a collection of sets of attribute instantiations such that each set contains at most one value for each attribute and such that F is closed under $\subseteq$, *then T is an extended SE-tree for F iff:*

1. *The root of T is labeled by the empty set;*
2. *The children of a node S in T are*

$$\{S \cup \{(A{=}v)\} \in F \mid A \in View(ind,S),\ v \in Dom(A)\}.$$

Figure 6 depicts an extended SE-tree for the complete space defined by three boolean attributes. Note the use of reduced notation where i stands for $\{A_i = T\}$, and $-i$ represents $\{A_i = F\}$.

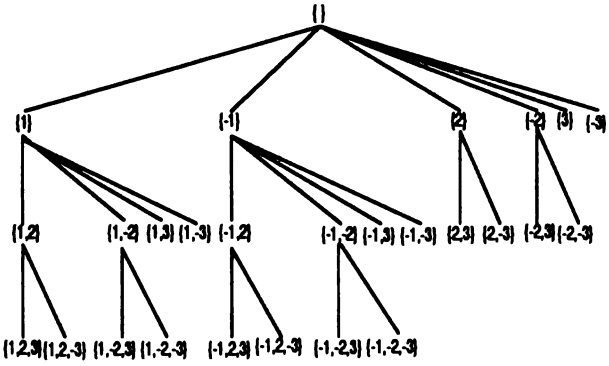


Figure 6: Complete SE-tree for 3 Boolean Attributes

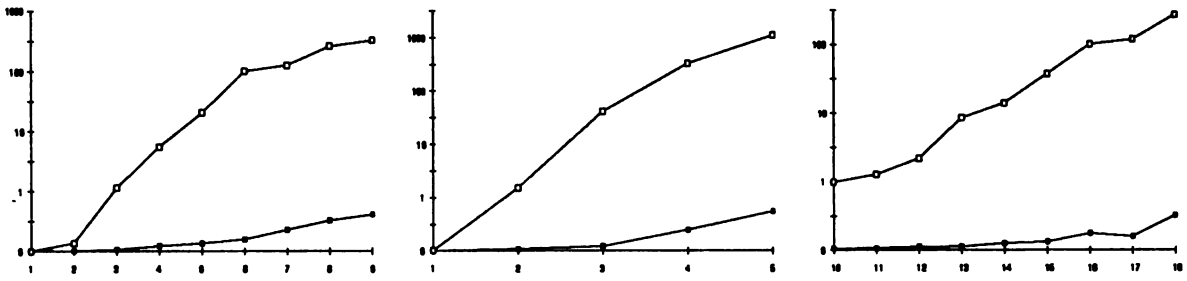


Figure 4: Run Time of SE-HS (■) versus HS-dag (□)

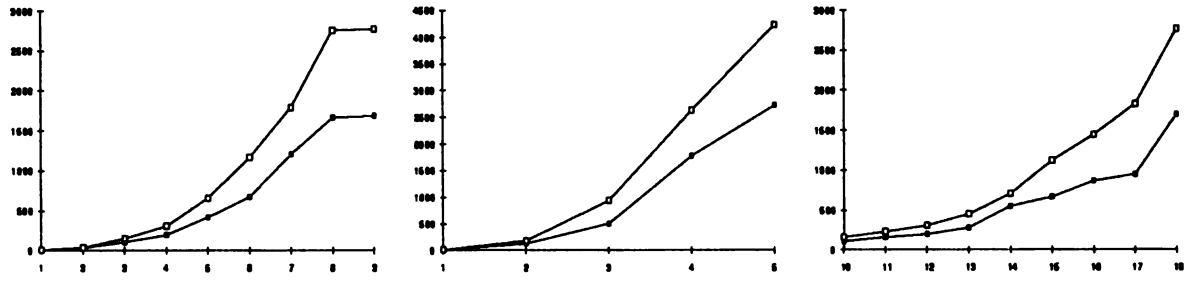


Figure 5: Number of Nodes Explored by SE-HS (■) versus HS-dag (□)

5 SE-TREE-BASED PRIME IMPLICATE ALGORITHM

In this section, we present an extension of SE-HS and demonstrate its use for the diagnostic framework of [de Kleer et al. 90]. We begin with a short description of the extended theory where kernel diagnoses are characterized as prime implicants of the (newly defined) set of conflicts. An extension of SE-HS, presented next, can be used to find those kernel diagnoses. The extended SE-HS has other useful properties: it can be flexibly focused; it can work with multiple behavioral modes; and it has an incremental version. A two-mode restriction of this algorithm corresponds to an old prime implicate generation algorithm [Slagle et al. 70]. We first demonstrate the empirical performance of this restricted version compared to a recent prime implicate generation algorithm. We then augment it with a new problem decomposition tactic, thereby obtaining an improved algorithm for prime implicate generation.

5.1 EXTENDED THEORY OF DIAGNOSIS

[de Kleer et al. 90] extended Reiter's theory with the notion of kernel diagnoses. Rather than having a diagnosis represent only faulty components (with the implicit assumption that all other components function properly), the new theory allows a diagnosis to explicitly specify working *and* non-working condition, without any presumption about other components' state.

Definition 5.1 AB-Clause [de Kleer et al. 90]

Let an AB*-literal be* AB*(c), or* ¬AB*(c) for some* c∈COMPS. *An* AB*-clause is a disjunction of* AB*-literals containing no complementary pair of* AB*-literals. An* AB*-clause is positive if all its* AB*-literals are positive.*

Definition 5.2 Conflict [de Kleer et al. 90]

A conflict is any AB*-clause entailed by SD∪OBS. A conflict set is its underlying set of* AB*-literals.*

Note that the new definition extends Reiter's original definition which, roughly speaking, allows only *positive* conflicts. We shall interchangeably speak about conflicts and their underlying sets.

Definition 5.3 Partial Diagnosis [de Kleer et al. 90]

A partial diagnosis is a conjunction of AB*-literals P such that P is satisfiable (does not contain complementary pairs), and for any other satisfiable conjunction ϕ covered by P, SD∪OBS∪ϕ is satisfiable.*

In other words, not only is P consistent with the system description and the observed behavior, but also any extension of P that assigns either AB, or ¬AB to components not mentioned in P, is also consistent.

Definition 5.4 Kernel Diagnosis [de Kleer et al. 90]

A kernel diagnosis is a partial diagnosis such that the only partial diagnosis which covers it is itself.

[de Kleer et al. 90] use the notion of prime implicants to characterize kernel diagnoses:

Definition 5.5 Prime Implicant [de Kleer et al. 90]

A conjunction π of AB*-literals, containing no complementary pairs, is an implicant of SD∪OBS if it entails every formula in SD∪OBS. It is a prime implicant if it is not covered by any other implicant.*

Theorem 5.6 [de Kleer et al. 90] *The kernel diagnoses are precisely the prime implicants of SD∪OBS.*

There are several early algorithms for computing prime implicants (or prime implicates)[1], used primarily for Boolean minimization (e.g. [Tison 67, Slagle et al. 70]). Recent interest in the AI community, for tasks such as ATMS encoding and circumscription, has yielded new algorithms (e.g. [Ngair 92]) as well as improvements to old algorithms (e.g. [Kean & Tsiknis 90, de Kleer 92]). Next, an extension of SE-HS will be shown to find kernel diagnoses, and therefore to generate all prime implicants of a CNF formula.

5.2 SE-HS EXTENDED

[de Kleer et al. 90] characterize kernel diagnoses as the prime implicants of SD∪OBS. Alternatively, kernel diagnoses can be defined in terms of hitting sets.

Theorem 5.7 Kernel Diagnoses and Conflicts

Let CONFLICTS *be the collection of conflict sets. The kernel diagnoses are precisely those minimal hitting sets for* CONFLICTS *that do not contain complementary pairs of* AB*-literals.*

Two important implications are (a) that SE-HS can be modified to find kernel diagnoses, and (b) that the modified algorithm can also serve to find prime implicants (implicates) in other settings. The proof for an extended version of this theorem can be found in [Rymon 92a]. Algorithm 5.8 presents the extended version of SE-HS's *Expand* procedure; the main program remains as previously described.

[1] Prime implicates are the disjunctive counterparts of prime implicants. Although for the purpose of diagnosis, we will be interested in prime implicants, most algorithms can compute both.

Algorithm 5.8 Node Expansion (version 3)

Procedure Expand(S)

1. *Let Window(S)←*
   $\{ c \mid c \in View(ind,S) \} \cap$
   $\{ c \mid ind(c) \leq \min_{S' \in NYH(S)} \max_{c' \text{ appears in } S'} ind(c') \}$
2. *For each $c \in Window(S)$ for which there exists some $B \in \{$*AB*, ¬*AB*$\}$ such that B(c) participates in some set from NYH(S) do*
3. *Unless there is $S' \in HS$ such that $S' \subseteq S \cup \{B(c)\}$*
4. *If $S \cup \{B(c)\}$ is a hitting set, add it to HS;*
5. *Otherwise, add it to OPEN-NODES.*

The new *Expand* procedure assigns state (AB or ¬AB) to a new component, not yet in the expanded set. The algorithm's correctness is easy to verify.

Besides its simplicity, being derived from a general SE-tree-based framework, SE-HS enjoys the following features:

1. *Focusing facility.* Due to the possibly overwhelming number of hypothetical diagnoses, much research on ATMS-based diagnostic programs has centered on methods for focusing on the most probable solutions (e.g. [Forbus & de Kleer 1988, de Kleer 91]). [Provan & Poole 91] advocate a preference criterion that is based on a diagnosis' *use*. Exploration policies, as in Section 3.5, can be used for that purpose.
2. *Fault models.* The importance of explicit models of faulty behavior has been recognized in the model-based diagnosis community (e.g. [Holzblatt 88, de Kleer & Williams 89]). In [Rymon 92a], we extend the diagnostic theory of [de Kleer et al. 90] to multiple behavioral modes and prove that kernel diagnosis in the new theory can still be characterized in terms of hitting sets. SE-HS can be easily extended to any number of behavioral modes.
3. *Incrementalism.* [Rymon 92a] outlines an incremental diagnostic framework that is based on a variation of SE-HS which can incrementally refine its hypothesis as conflicts arrive.

5.3 PERFORMANCE EVALUATION

We have implemented the extended SE-HS algorithm and have compared its performance to that of a PHI-based prime implicate generation algorithm [Ngair 92]. As before, the two algorithms were run on hundreds of examples that were randomly generated according to the three parameters (#conf, #lit, #comp). Due to the relative strength of both algorithms, we used larger examples in this experiment. As a side note,

the SE-HS implementation is general in that it can take any number of behavioral modes. This generality is not useful in the experiment, where examples are bi-modal. Figure 7 depicts two one-way sensitivity analyses (for #conf, #lit) and one three-way analysis. Again, shadowed squares correspond to SE-HS performance, open ones to that of the PHI-based algorithm.

5.4 PROBLEM DECOMPOSITION

As so far presented, we could draw a correspondence between nodes explored by the bi-modal version of the extended SE-HS algorithm and the operation of an old prime implicate generation algorithm proposed by [Slagle et al. 70]. This is important for two reasons: first, it reveals the general SE-tree-based features of Slagle et al.'s algorithm, but more importantly, our next improvement to SE-HS will result in an improved version of their algorithm.

Where feasible, problem decomposition (also referred to as divide-and-conquer) is a well known strategy to sharply reduce problem solving costs (time, space, etc.) In the context of diagnosis, such an opportunity may arise when a fault is composed of a number of unrelated, or partially related sub-faults. [Wu 90] shows tremendous gain in utilizing problem decomposition techniques in diagnosis.

In the context of multiple fault diagnosis, in addition to potential saving of time and space, decomposition may also lead to more compact *representation* of a solution. In many cases, a solution can be *written* more compactly if it is factored. For example, a solution of the form $\times_{i=1}^{n}$ { AB(c_{2i-1}), AB(c_{2i}) }, when expanded, consists of 2^n minimal diagnoses. Put differently, like formulae, some solutions can be represented compactly as CNF whereas others are more concise in their disjunctive form. This is, roughly, the intuition behind the following heuristic.

Theorem 5.9 Problem Decomposition

If CONFLICTS *can be partitioned into two disjoint subsets C' and C", such that no component appears in both subsets, then the minimal hitting sets (MHS) for* CONFLICTS *are given by:*

$$\text{MHS(CONFLICTS)} = \text{MHS(C')} \times \text{MHS(C")}$$

If a partition exists, it can clearly save significant work. Recursive application of SE-HS to each of the two partitions can cut the exponential search space into two smaller search spaces. The notion of partitioning can be extended to *any* number of partitions, making the latter equivalence classes and making the partitioning *unique*. The solution in such case is the *Cartesian product* of the sub-solutions.

Fortunately, if one exists, there is a simple, almost-linear, algorithm that finds a partitioning for a collection of sets (e.g., using a union-find strategy [Tarjan 83]). Moreover, even if there is no facilitating partitioning to begin with, it is possible that one exists when a node's particular view is considered. Given a node S, recall that any of S's descendants will only expand with respect to $View(ind, S)$. Thus, it is enough to look for a partition in the restriction of $NYH(S)$ to $View(ind, S)$.

Algorithm 5.10 An Amendment to Expand

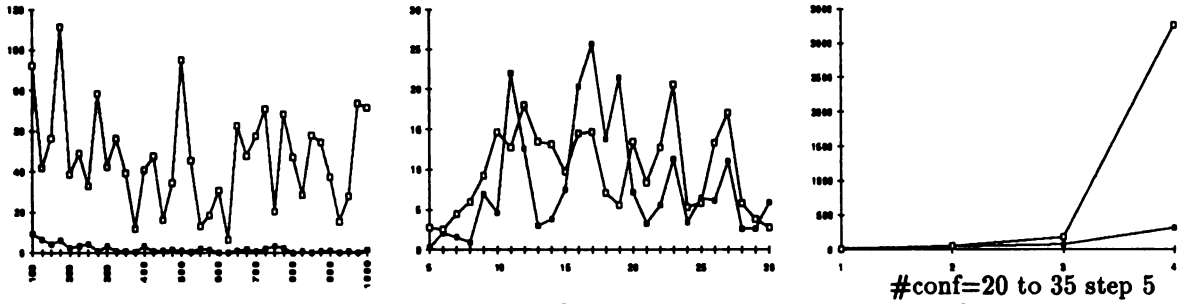
1. *Let* Γ *be the restriction of NYH(S) to components in View(ind,S).*
2. *If there is a partitioning* $\Gamma = \times_{i=1}^{k} \Gamma_i$ *then*
3. *Run SE-HS on each of the* Γ_i *independently. Let Hitting*(Γ_i) *be the corresponding results, merge* $\{S\} \times (\times_{i=1}^{k}$ *Hitting*(Γ_i)) *into HS while checking for possible subsumption.*
4. *Otherwise, expand S as usual.*

Exact prioritization is a problem in the augmented algorithm since every node in a new tree represents only part of (possibly many) solutions. For similar reasons, subsumption has to be more aggressively monitored (although this is easily done when hitting sets are cached in an SE-tree-based data structure). Before, subsumption was avoided by the subsuming solution being discovered prior to the subsumed one. Now, it is possible that a solution node in the original SE-tree will be subsumed by some but not all of the solutions in which a given node in some new tree participates. Nevertheless, problem decomposition is still attractive since it is particularly effective in problems which admit highly disjunctive solutions. Those are hardest for the original SE-HS algorithm. The following example demonstrates the effectiveness of the problem decomposition heuristic.

Example 5.11 Consider the following collection of conflicts: { {AB(1),¬AB(3),AB(4)}, {AB(5),AB(6)}, {AB(3),AB(4)}, {AB(2),¬AB(6)}}. Figure 8 illustrates the SE-tree explored by SE-HS *without* decomposition. As before, O's denote hitting sets, X's mark closed nodes. Exploration for the same problem *with* decomposition is depicted in Figure 9. There, the first step involved partitioning the collection of conflicts into two disjoint sets. Thereafter, two sub-problems are solved, and the solution is the cross-product of the respective results, i.e. {{AB(1),AB(3)}, {AB(4)}} × {{AB(2),AB(5)}, {AB(2),AB(6)}, {AB(5),¬AB(6)}}. The reductions in time and space are obvious.

5.5 ABDUCTIVE DIAGNOSTIC MODELS

In [Reggia et al. 85], diagnosis is formulated as a generalized set covering (GSC) problem. In their basic



#conf=100 to 1000 step 25 | #lit=5 to 30 | #conf=20 to 35 step 5, #lit=12 to 15, #comp=12 to 15

Figure 7: Run Time of extended SE-HS (■) versus PHI-based algorithm (□)

Figure 8: SE-tree without Decomposition

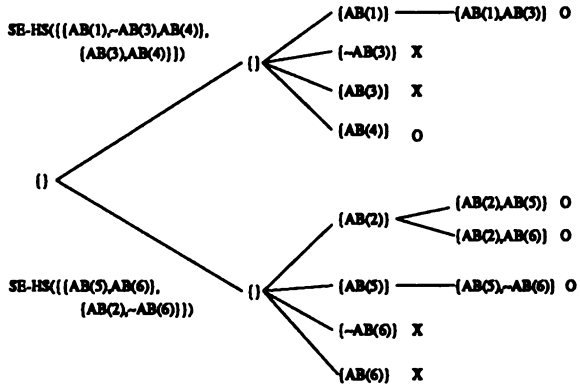


Figure 9: SE-tree Exploration with Decomposition

model, a diagnostic problem is represented in a bipartite graph in which symptoms and disorders form each of the respective partitions. Each disorder in the graph is linked to all of its symptoms via a *causes* relation. Given a set of observed symptoms, a diagnosis is defined as a minimal set of disorders which covers all symptoms.

A most-probable-first search algorithm for that problem is described in [Peng & Reggia 87]. It searches the space of sets of disorders for such sets which cover all symptoms. This algorithm, however, is redundant in that partial hypotheses may be discovered repeatedly during search. That redundancy could be avoided if an SE-tree framework were adopted.

Alternatively, the problem can be turned into a hitting set problem. [Reiter 87] presents a transformation of a GSC representation of a diagnostic problem into his own framework. There is, in fact, a better transformation which avoids the conflict generation part of Reiter's theory by mapping the GSC problem directly into an HS one. Then, we could simply use SE-HS. Given a set of symptoms s_i, we could define a "conflict set" for each symptom:

$$\text{conflict}(s_i) \stackrel{\text{def}}{=} \{d \mid d \text{ is a disease, } d \textit{ causes } s_i\}$$

Presented with s_i, the conflict asserts that it is impossible that none of its causing disorders are present. It is easy to prove that a set of disorders is a minimal set cover iff it is a minimal hitting set for such conflicts.

In [Peng & Reggia 87], hypotheses are explored by their likelihood. The SE-tree-based framework allows such exploration, as well as a variety of other exploration policies. In [Peng & Reggia 87], non-minimal hypotheses are also explored. This is easily done in SE-HS by removing the subsumption requirement (*Expand*, step 3). In addition, pruning rules (cf. Section 3.6) can be used to avoid unpromising parts of the search space. Problem decomposition (cf. Section 5.4) may also be helpful in reducing time and cost. Finally, it seems that other models of diagno-

sis in which solutions are defined in terms of sets, e.g. [Bylander et al. 91, Poole 91, Console & Torasso 91], can also use an SE-tree-based search framework in their implementations.

6 LEARNING MINIMAL CLASSIFICATION RULES

Decision trees are an important tool, and serve as an underlying representation in many problem solving tasks. Significant research in Machine Learning has used decision trees in architectures for induction of classification knowledge from examples. Best known are ID3 [Quinlan 86] and its descendants. In [Rymon 92b], we present an SE-tree-based characterization of the induction task, contrast it from classification and search perspectives with the decision-tree-based framework, and compare the two empirically. Here, we will only contrast features of the two representations, concentrating on search aspects.

Definition 6.1 Rules

A training set (TSET) is a collection of examples. Each example is a complete description for which a correct classification (denoted π) is known. A rule is a partial description R such that if $t,t' \in TSET$ are such that $R \subseteq t,t'$, then $\pi(t) = \pi(t')$. It is minimal if none of its subsets is a rule.

The objective of a learning system is to learn rules that can be expected to perform well not only on the training set, but also on new examples. While there is no consensus as to the precise composition of such a collection, it is fairly acceptable that *general* (minimal) rules are preferable to specific ones. We shall therefore concentrate on finding minimal classification rules[2].

6.1 PROPOSED SOLUTION

ID3 constructs a decision tree in which internal nodes are labeled with attributes, edges with instantiations of these attributes, and leaves with a class prediction. Briefly, the tree is constructed by successively partitioning the set of training examples until all remaining examples are equally classified. Such node becomes a leaf and is labeled with that class.

While construction of an arbitrary decision tree that correctly classifies the training data is straightforward, it is well known that the success of decision-tree-based algorithms on future data is crucially dependent on the particular order in which the attributes were chosen in the successive refinement steps [Fayyad & Irani 88, Goodman & Smyth 88]. As Quinlan notes, one can often not afford to generate all possible decision trees in order to choose the best one. Thus, ID3 (as do other algorithms) uses a heuristic to guide its choice of attributes. One prominent heuristic is based on entropy-minimization, using Shannon's information-theoretic measure.

6.2 SE-TREE-BASED ALTERNATIVE

Aimed at *all minimal* rules, SE-Learn (Algorithm 6.3) uses an SE-tree-based framework. As before, open nodes are prioritized, facilitating various exploration policies. In the context of learning, these will be used to represent *bias* and will be briefly discussed in the end of this section. As before, SE-Learn exploits the systematic ordering to prune away unpromising parts of the search space (i.e. nodes which cannot lead to minimal rules).

Definition 6.2 Candidate Expansions

Let S be a node, $TSET(S) \stackrel{\text{def}}{=} \{t \in TSET \mid S \subseteq t\}$. We say that $(A=v)$ is a candidate expansion of S if $A \in View(ind,S)$, $v \in Dom(A)$, and in addition $TSET(S \cup \{(A=v)\}) \neq TSET(S)$. A node S will be called impotent if either (1) TSET(S) is empty; or (2) there exist $t,t' \in TSET(S)$ disagreeing on their class, and only differing in their assignment to attributes not in View(ind,S).

Algorithm 6.3 Induction of Minimal Rules

Program SE-Learn (TSET)

1. *Let RULES* $\leftarrow \{\}$, *OPEN-NODES* $\leftarrow \{\{\}\}$
2. *Until OPEN-NODES is empty do*
3. *Expand(Next-Best(OPEN-NODES))*

Procedure Expand(S)

1. *For each candidate expansion $(A=v)$, let* $R \stackrel{\text{def}}{=} S \cup \{(A=v)\}$, *do*
2. *If R is not impotent, nor is it subsumed by any $R' \in RULES$ then*
3. *If R is a rule then add it to RULES;*
4. *Or else add it to OPEN-NODES.*

Theorem 6.4 *If open nodes are prioritized by their label's cardinality then SE-Learn is correct (produces all and only minimal rules.)*

Given the incompleteness of the examples with which they are presented, learning programs may often have to choose among a number of candidate classifiers, all of which are consistent with the training

[2] We simplify here. Motivation for learning *all* minimal rules, variations of an SE-tree-based algorithm that learn subsets of this collection, and empirical results are given in [Rymon 92b].

set. External preference criteria, also referred to as *bias* [Mitchell 80], may be necessary for that purpose. Within an SE-tree-based framework, exploration policies can serve in the implementation of such bias. In programs such as SE-Learn, where *all* rules are explored during the learning phase, an exploration policy will serve in the classification of new objects by guiding preference over possibly conflicting rules. In variants of SE-Learn in which only a subset of the rules are learned, an exploration policy will implement a preference among possible subsets. As was the case for SE-HS, any exploration policy that is monotonic will result in a correct algorithm. Important policies include (1) exploration by *cardinality*, where a preference is given to simpler rules; (2) by *probability* (using either a known distribution or frequency in the training set), resulting in preference to characterization of denser parts of the search space; (3) using *Shannon's information-theoretic measure*, preferring more discriminating rules; and (4) by *utility* or some other monotone preference criterion.

6.3 PROJECTED GAIN AND COST

Three related problems arise when a decision tree is used as a framework for search and representation of minimal rules:

1. *The minimality problem – rules will often not be discovered in their minimal form;*
2. *The multiplicity problem – a minimal rule may be discovered repeatedly, disguised in a number of its minimal subsets; and*
3. *The incompleteness problem – some minimal rules may not be discovered at all.*

The minimality problem is often addressed by subsequently pruning the rules extracted from the decision tree [Quinlan 87]. The replication problem, a special case of multiplicity in which sub-trees are replicated within a single decision tree, has been addressed by several researchers, e.g. [Rivest 87, Pagallo & Haussler 90]. The more general multiplicity problem, however, may take many other forms. Incompleteness is the result of the mutual exclusiveness property of decision-tree-based rules (see [Weiss & Indurkhya 91]).

In contrast, the SE-tree-based framework does not suffer from these problems:

1. *Rules are always discovered in minimal form;*
2. *Minimal rules are always discovered uniquely; and*
3. *All minimal rules are discovered.*

The fact that *any* given decision tree may suffer from those problems suggests that none is globally optimal. The SE-tree, however, can be shown to embed many decision trees. More specifically, *all* decision trees in which attributes are chosen monotonically with respect to some arbitrary indexing, are topologically and semantically equivalent to a tree formed from a subset of the SE-tree's edges.

Complexity-wise, the SE-tree's exhaustiveness and relatively large *initial* branching factor are deceiving. Its complexity is fairly close to that of a *single* decision tree:

Theorem 6.5 SE-Tree Size

If all attributes are b-valued, then the number of nodes in a complete decision tree is $\Sigma_{i=0}^{n} b^i > b^n$. *In sharp contrast, the size of a super-tree in which all decision trees are embedded is significantly larger:* $b^n \cdot n!$. *The size of a complete SE-tree is only* $(b+1)^n$.

7 Summary

Many problems in which partial sets or partially instantiated set of variables are targeted share a common structure when viewed as search problems. We presented the *Set-Enumeration* (SE)-tree as a simple, complete and irredundant vehicle for representing and/or enumerating sets in a best-first fashion. As such, it can serve as the basis for a search-based framework for many such problems.

To demonstrate its usefulness and effectiveness, we presented SE-tree-based algorithms for the hitting-set problem, in the context of consistency-based diagnosis. We used the particular instantiations of these algorithms to demonstrate general features of the paradigm, and compare it with current algorithms. Throughout this process, we developed several add-on tactics including SE-tree-based pruning rules, exploration policies, and problem decomposition methods. Besides their particular incarnations in the SE-HS algorithms, those methods are general and can be shared across many problem domains. In the last part of this paper, in the context of rule induction, we compared features of an SE-tree-based representation with one that is based on decision trees.

Acknowledgements

This research was supported in part by a graduate fellowship ARO Grant DAAL03-89-C0031PRI. I thank Teow-Hin Ngair, Greg Provan, Russ Greiner, Alex Kean, and Ron Rivest for useful discussions and suggestions. I also thank Kevin Atteson, Michael Niv, Philip Resnik, Jeff Siskind, and Bonnie Webber for comments on previous drafts. Finally, I am grateful to Barbara Smith for providing the HS-dag implementation and Teow-Hin Ngair for providing the code of his prime implicate generation algorithm.

References

[Bylander et al. 91] Bylander, T., Allemang, D., Tanner, M. C., and Josephson, J., The Computational Complexity of Abduction. *Artificial Intelligence* 49, pp. 25-60, 1991.

[Console & Torasso 91] Console, L., and Torasso, P., A Spectrum of Definitions of Model-Based Diagnosis. *Computational Intelligence*, 7, pp. 133-141, 1991.

[de Kleer & Williams 89] de Kleer, J. and Williams, B., C., Diagnosis with Behavioral Modes. *Proc. IJCAI-89*, Detroit MI, pp. 1324-1330, 1989.

[de Kleer et al. 90] de Kleer, J., Mackworth, A. K., and Reiter, R., Characterizing Diagnoses. *Proc. AAAI-90*, pp. 324-330, Boston MA, 1990.

[de Kleer 91] de Kleer, J., Focusing on Probable Diagnoses. *Proc. AAAI-91*, pp. 842-848, Anaheim CA, 1991.

[de Kleer 92] de Kleer, J., An Improved Incremental Algorithm for Generating Prime Implicates. *Proc. AAAI-92*, pp. 780-785, San Jose CA, 1992.

[Fayyad & Irani 88] Fayyad, U. M., and Irani, K. B., What Should be Minimized in a Decision Tree? *Proc. AAAI-90*, pp. 749-754, Boston MA, 1990.

[Forbus & de Kleer 1988] Forbus, K. D., and de Kleer, J., Focusing the ATMS. *Proc. AAAI-88*, pp. 193-198, Saint Paul MN, 1988.

[Goodman & Smyth 88] Goodman, R., and Smyth, P., Decision Tree Design from a Communication Theory Standpoint. *IEEE Trans. on Information Theory*, 34(5), 1988.

[Greiner et al. 89] Greiner, R., Smith, B. A., and Wilkerson R. W., A Correction to the Algorithm in Reiter's Theory of Diagnosis. *Artificial Intelligence*, 41, pp. 79-88, 1989.

[Holzblatt 88] Holzblatt, L. J., Diagnosing Multiple Failures Using Knowledge of Component States. *Proc. IEEE AI Applications*, pp. 139-143, 1988.

[Karp 72] Karp, R. M., Reducibility Among Combinatorial Problems. In Miller and Thatcher eds., *Complexity of Computer Computations*, Plenum Press, New York, pp. 85-103, 1972.

[Kean & Tsiknis 90] Kean, A., and Tsiknis, G., An Incremental Method for Generating Prime implicants/Implicates. *Journal of Symbolic Computation*, 9:185-206, 1990.

[Knuth 73] Knuth, D. E., The Art of Computer Programming, Vol. 3: Sorting and Searching. *Addison Wesley*, 1973.

[Mitchell 80] Mitchell, T. M., The Need for Biases in Learning Generalizations. *TR 5-110*, Rutgers University, 1980.

[Ngair 92] Ngair, T., *Convex Spaces as an Order-Theoretic Basis for Problem Solving*, Ph. D. Thesis, Department of Computer and Information Science, University of Pennsylvania, 1992.

[Pagallo & Haussler 90] Pagallo, G., and Haussler, D., Boolean Feature Discovery in Empirical Learning. *Machine Learning*, 5, pp. 71-99, 1990.

[Peng & Reggia 87] Peng, Y., and Reggia, J. A., Being Comfortable with Plausible Diagnostic Hypotheses. *TR 1753*, University of Maryland, 1987.

[Poole 91] Poole, D., Representing Diagnostic Knowledge for Probabilistic Horn Abduction. *Proc. IJCAI-91*, Sydney, Australia, 1991.

[Provan & Poole 91] Provan, G. M., and Poole, D., The Utility of Consistency-Based Diagnostic Techniques. *Proc. KR-91*, Cambridge MA, pp. 461-472, 1991.

[Quinlan 86] Quinlan, J. R., Induction of Decision Trees. *Machine Learning*, 1(1):81-106, 1986.

[Quinlan 87] Quinlan, J. R., Generating Production Rules from Decision Trees. *Proc. IJCAI-87*, pp. 304-307, 1987.

[Reggia et al. 85] Reggia, J. A., Nau, D. S. and Wang, P. Y., A Formal Model of Diagnostic Inference. I. Problem Formulation and Decomposition. *Information Sciences*, 37, pp. 227-285, 1985.

[Reiter 87] Reiter, R., A Theory of Diagnosis From First Principles. *Artificial Intelligence*, 32, pp. 57-95, 1987.

[Rivest 87] Rivest, R., Learning Decision Lists. *Machine Learning*, 2, pp. 229-246, 1987.

[Rymon 92a] Rymon, R., SE-tree-based Candidate Generation: Systematic Exploration in Model-Based Diagnosis. *In preparation*, 1992.

[Rymon 92b] Rymon, R., An SE-tree-based Characterization of the Induction Problem. *In preparation*, 1992.

[Tison 67] Tison, P., Generalized Consensus Theory and Application to the Minimization of Boolean Functions. *IEEE Trans. on Computers*, 16(4):446-456, 1967.

[Slagle et al. 70] Slagle, J. R., Chang, C, and Lee R. C., A New Algorithm for Generating Prime Implicants. *IEEE Trans. on Computers*, 19(4), 1970.

[Tarjan 83] Tarjan, R., Data Structures and Network Algorithms. *Society for Industrial and Applied Mathematics*, Philadelphia PA, 1983.

[Weiss & Indurkhya 91] Weiss, S. M., and Indurkhya, N., Reduced Complexity Rule Induction. *Proc. IJCAI-91*, pp. 678-684, Sydney, Australia, 1991.

[Wu 90] Wu, T. D., A Problem Decomposition Method for Efficient Diagnosis and Interpretation of Multiple Disorders. *Proc. SCAMC-90*, pp. 86-92, Washington DC, 1990.

IX.

Nonmonotonic Logics

Maps Between Nonmonotonic and Conditional Logic

Horacio L. Arlo-Costa
Philosophy Dept.
Columbia University
New York, NY 10027
ha8@cunixb.cc.columbia.edu

Scott J. Shapiro
Philosophy Dept.
Columbia University
New York, NY 10027
ha8@cunixb.cc.columbia.edu

Abstract

The minimal conditional logics corresponding to the preferential systems $\mathcal{P}$ (Kraus-Lehmann-Magidor), $\mathcal{R}$ and $\mathcal{RR}$ (Lehmann-Magidor) are identified as the Generalized Horn-flat counterparts of the systems $\mathcal{B}$ (Burgess), $\mathcal{V}$ (Lewis) and $\mathcal{NP}$ (Delgrande), respectively. It is shown that the innovative use of states instead of possible worlds in preferential models is inessential for a complete and sound characterization of these systems. The map is finally compared with related works in the field.

1 INTRODUCTION

The syntactic similarities between the systems of conditional and nonmonotonic logic are apparent to anyone familiar with these two fields. It is striking, for example, that the classical inference patterns that fail in conditional logic, such as the strengthening of the antecedent, transitivity and contraposition, also fail at the metalevel when a deviant notion of consequence is introduced to represent commonsense reasoning. The failure of the strengthening rule for the conditional connective '>' is a clear counterpart of the nonmonotonicity of the notions of consequence used in nonmonotonic logic.

Although the counterfactual conditional '>' lacks some of the undesirable properties of '$\rightarrow$' to represent defaults, it is nonetheless too strong. The offending rule which is valid for counterfactuals but not for prima facie inferences can be expressed formally as: (MP) $(A > B) \rightarrow (A \rightarrow B)$. Obviously a formal system of conditional logic having this rule and the classical modus ponens would allow for the detachment of B in the presence of A, given $A > B$.

This result is unacceptable for those who wish to use the '>' to construct a purely logical framework for the representation of defaults. In fact, researchers interested in the construction of such a framework, such as James Delgrande in [9], have developed formal treatments of the connective '>' that allow for formulas of the following shape to be jointly satisfiable while having true antecedents.

(1) Raven(Pete) > Black(Pete)
(2) Raven(Pete) $\wedge$ Albino(Pete) > $\neg$ Black(Pete)

It is clear that in a logic which has the rule (MP) for '>', the joint assertion of (1) and (2) together with their antecedents will lead to inconsistency. Therefore, one of the saliences that characterise the conditional systems used for representational purposes in AI is that they characteristically lack some of the axioms usually associated with the counterfactual interpretation of conditionals, like (MP). As we will see they coincide formally with conditionals traditionally associated with a deontic interpretation.

In the last five years, several researchers have tried to build a uniform approach to nonmonotonic logic without appealling to conditional logic. Shoham [19], Gabbay [10], Makinson [17], Kraus, et al. [13], and Lehmann, et al. [15], among others, have developed both syntactical systems (usually expressed at the level of the metalanguage using a Gentzen-style sequent operator) and a possible-worlds semantics for these unified nonmonotonic calculi which reflect the general patterns used in commonsense reasoning.

Nevertheless, the interest in exploring the relationships between the conditional-based approaches and the systems reflecting the general patterns used in everyday nonmonotonic reasoning is very recent. In particular, a general and exact characterisation of the conditional systems corresponding to the family of preferential systems studied by Kraus, et al., and Lehmann, et al., is lacking in the literature. In this article we will provide such a characterisation.

2 CONDITIONAL LOGIC

In this section, we will review several systems of conditional logic that have been proposed by researchers in the computer science and philosophical logic communities. Our task will be somewhat complicated not only because of the various axiomatizations extant in the literature, but also by the variety of semantical characterizations used by these conditional logicians. The systems discussed vary not only on the level of model construction but also on the level of truth definitions.

This overview will have both a syntactic and semantic agenda. On the syntactic side, we will review the various axiomatic structures of the more popular systems of conditional logic and, then, define a new class of conditional logics by considering the Generalised Horn-flat counterpart of these existing systems. Semantically, we review some of the formal features of the conditional models with which we are dealing and demonstrate that, at the level of model structure, James Delgrande's conditional system $\mathcal{NP}$ can be represented using a system of spheres semantics. Finally, we develop and present a unified framework in which all of the conditional systems can be compared.

2.1 Syntax

The Conditional Logic $\mathcal{NP}$

The conditional logic $\mathcal{NP}$, proposed by James Delgrande in [9], is the smallest logic containing the standard propositional calculus and closed under the following axiom schemata and rule of inference:

ID $A > A$

CC $((A > B) \wedge (A > C)) \rightarrow (A > (B \wedge C))$

RT $(A > B) \rightarrow (((A \wedge B) > C) \rightarrow (A > C))$

CV $\neg(A > B) \rightarrow ((A > C) \rightarrow ((A \wedge \neg B) > C))$

CC' $((A > C) \wedge (B > C)) \rightarrow ((A \vee B) > C)$

RCM If $\vdash B \rightarrow C$ then $\vdash (A > B) \rightarrow (A > C)$

The Conditional Logic $\mathcal{V}$

The conditional logic $\mathcal{V}$ of David Lewis[16] is the smallest logic containing the propositional calculus, closed under **ID**, **CC**, **CC'**, **CV**, **RCM** and the following axiom:[1]

CSO $((A > B) \wedge (B > A)) \rightarrow ((A > C) \leftrightarrow (B > C))$

Observation 2.1 *The system obtained by adding the axiom* **CM**: $((A > B) \wedge (A > C)) \rightarrow ((A \wedge B) > C)$ *to the system $\mathcal{NP}$ is equivalent to the system $\mathcal{V}$.*

[1] This axiomatisation is due to Katsuno-Satoh[12].

The Conditional Logic $\mathcal{SS}^*$

The conditional logic $\mathcal{SS}^*$ is the smallest logic containing the propositional calculus and closed under **ID**, **CC**, **CC'**, **CM**, **RCM**, and the following rule of inference:

RCEA If $\vdash B \leftrightarrow C$ then $\vdash (A > C) \leftrightarrow (B > C)$

Observation 2.2 *The system $\mathcal{SS}^*$ is obtained by subtracting* **CV** *from the system $\mathcal{V}$.*

Observation 2.3 *The system $\mathcal{SS}^*$ is equivalent to the system B presented by Burgess in [8].*

The system $\mathcal{SS}^*$ is a weaker version of the well-known system $\mathcal{SS}$ studied by John Pollock in [18]. In order to obtain the system of Pollock, it is sufficient to add the following axioms to $\mathcal{SS}^*$:

MP $(A > B) \rightarrow (A \rightarrow B)$

AS $(A \wedge B) \rightarrow (A > B)$

The systems $\mathcal{SS}^*$, $\mathcal{V}$ and $\mathcal{NP}$ will be the minimal conditional systems that we will correlate with the preferential systems of Kraus, et al. and Lehmann, et al. Notice that $\mathcal{SS}^*$ and $\mathcal{NP}$ are weaker versions of $\mathcal{V}$. The first is obtained by dropping **CV** from $\mathcal{V}$ and the second is obtained by dropping **CM** from $\mathcal{V}$. In the section following the next, we will propose semantics for the formal systems presented here and we will clarify the relations among $\mathcal{SS}^*$, $\mathcal{V}$ and $\mathcal{NP}$ (and some enriched versions of them) under a semantic point of view.

Absolute and Universal Extensions

The logics $\mathcal{SS}^*$ and $\mathcal{V}$ can be extended to $\mathcal{SS}^*\mathcal{TA}$ and $\mathcal{VTA}$ by the addition of the following two axioms:

T $\Box A \rightarrow A$

A $\Diamond(A \vee B) \wedge (A \vee B > \neg B) \rightarrow \Box(\Diamond(A \vee B) \wedge (A \vee B > \neg B))$

where the necessity operator is translated to the conditional as follows: $\Box A \equiv \neg A > A$.

Notice that we have not specified any absolute and universal extensions for the system $\mathcal{NP}$. As we will see in the section on semantics, $\neg A > A$ is not an appropriate interpretation of the $\Box$ for the $\mathcal{NP}$ family of systems. Hence, any attempt to translate the (T) axiom with conditionals as is done above will not be valid in $\mathcal{NP}$ systems.

2.2 Generalized Horn-flat Conditional Systems

In this section, we will define an interesting family of conditional logics, obtained by restricting the

standard conditional language and converting the axiomatic structure of the above formalism to a Gentzen-style deduction system whose rules of inference are constrained to follow a certain pattern ('Generalised Horn-flat'). As we will show later in the paper, these Generalised Horn-flat counterpart systems are the conditional logics that can be mapped to the nonmonotonic formalisms of Kraus, et al. and Lehmann, et al.

We will assume a basic stock of propositional variables $p, q, r, \ldots$, as well as the classical connectives and the sign '$>$'. $A, B, \ldots$, are the schemata-variables ranging over the formulas of the language. Also, we will take the wff$_{\rightarrow}$ to be the standard well-formed formulas of the propositional calculus.

Definition 2.1 1) If A and B are wff$_{\rightarrow}$'s, then $A > B$ is a wff$_{>}$
2) If A is a wff$_{>}$, then $\neg A$ is a wff$_{>}$

Definition 2.1 provides a specification of the non-nested conditional fragment of the language. Any boolean combination of sentences, aside from the negation of a wff$_{>}$, in which the sign $>$ appears is forbidden. We will call the 'non-boolean flat fragment' of any conditional logic the fragment characterised by Definition 2.1; 'flat', because clause (2) of Definition 2.1 eliminates the possibility of having nested conditionals, and 'non-boolean', because the definitions discard the possibility of having any boolean combination of conditional sentences or any combination of conditional and classical sentences, such as $(\neg A > A) \rightarrow A$.

The patterns of the shape: $((A_1 > B_1), \ldots, (A_n > B_n)) \vdash (A > B)$, where all expressions belong to wff$_{>}$, are usually called 'Horn-flat' [hereinafter HF] in the literature. Since we will allow negations in the left hand side (but no any other boolean combination of conditional sentences and their negations) we call these new patterns 'Generalised Horn-flat' [hereinafter GHF].

It is a trivial exercise to reformulate the conditional systems presented above in a Gentzen-style system, such that each axiom is transformed into a rule of inference of the form: $((A_1 > B_1), \ldots, \neg(A_i > B_i), \ldots, (A_n > B_n)) \vdash (A > B)$ (e.g., the axiom schema **CM** becomes $A > B, A > C \vdash A \wedge B > C$). We will also allow within these systems any contraposition of a GHF pattern. Therefore, theses with a negation of a wff$_{>}$ on the right-hand side of the '$\vdash$' can be derived in a counterpart system just in case they are equivalent to some GHF pattern. The importance of the GHF systems of conditional logic will be immediate for the reader after seeing, in Section 3, the syntax of the preferential systems.

2.3 System of Spheres Semantics

In this section, we will introduce a unified semantics for the systems $\mathcal{V}$ and $\mathcal{NP}$, utilising the system of spheres semantics developed by David Lewis in [16].

Definition 2.2 A spheres model is a double $\langle I, \$ \rangle$, where I is a non-empty set of worlds, $\$$ a function (called a system of spheres) that assigns to each world $i \in I$, a nested set $\$_i$ of subsets of I.

Definition 2.3 A valuation for $\langle I, \$ \rangle$ is a map P that assigns to each variable a subset of worlds in I. P is classical for the non-conditional fragment, and can be extended for the conditional fragment as follows: $P(A > B)$ is the set of worlds of all $i \in I$ such that either:

1) no A-world belongs to any sphere S in $\$_i$
or
2) $A \rightarrow B$ holds at every world in the smallest A-permitting sphere in $\$_i$

Definition 2.4 For any given spheres model S, $\models_S A > B$ iff $I = P(A > B)$.

A sentence or sentence schema is valid unconditionally if it is valid in every spheres model of the kind specified above. Nevertheless, certain constraints can be imposed over the spheres models. In this case, we will say that a sentence is valid under a combination of conditions iff it is valid under every spheres interpretation that satisfies these conditions.

The following is a list of constraints that will be used later on in the paper:

(L) Limit Assumption $\forall A \in L \forall i \in I$, if the proposition corresponding to A cuts $\bigcup \$_i$, then there is some smallest member of $\bigcup \$_i$ that overlaps the proposition A

(C) Centering $\forall i \in I, \{i\} \in \$_i$

(T) Total Reflexivity $\forall i \in I, i \epsilon \bigcup \$_i$

(A) Absoluteness $\forall i, j \in I, \$_i = \$_j$

(U) Uniformity $\forall i, j \in I, \bigcup \$_i = \bigcup \$_j$

(UT) Universality $\forall i \in I, \bigcup \$_i = I$

The system $\mathcal{V}$ presented above is complete and sound with respect to the class of system of spheres models constrained by (L). Notice that the models used by Lewis in [16] to construct his official semantics for $\mathcal{V}$ are different from the models used here. Lewis does not need to impose an (L) requirement on his models because he uses a different truth definition than the one given above. The system $\mathcal{V}$ is sound and complete with respect to 'pure' spheres models (without the constraint (L)) if the valuation P is extended with the alternative truth definition given by Lewis. Nevertheless, Lewis proves in [16] that when one uses Definition 2.3 as one's truth definition with (L) models, the class of valid sentences that arise from this mixture is equivalent to the the use of his truth definition with models that do not necessarily obey the limit assumption. Hence, we are justified in using the above truth

definition with the (L) models throughout this paper and referring to this system as Lewis' $\mathcal{V}$.

Unfortunately, the spheres-type of semantics is not universally used in the literature for model construction. For example, the semantics of the system $\mathcal{NP}$ is given in terms of a selection-function. We will first explain the basic facts about this semantics and then present an alternative semantics for $\mathcal{NP}$ using systems of spheres.

Definition 2.5 A selection-function model is a double $\langle W, f\rangle$, where W is a set of possible worlds, f is a function $f : W \times V(W) \rightarrow V(W)$, where V is a valuation, $V : L \rightarrow 2^W$. A classical V can be extended in the following manner:

> $V(A > B)$ is the set of worlds such that for all $i \in I$ $f(i, V(A)) \subseteq V(B)$.

In the case of the system $\mathcal{NP}$, f is constrained by the following set of conditions:

i) $f(w, V(A)) \subseteq V(A)$

ii) If $f(w, V(A)) \subseteq V(B)$, then $f(w, V(A)) \subseteq f(w, V(A \wedge B))$

iii) If $f(w, V(A)) \not\subseteq V(B)$, then $f(w, V(A \wedge \neg B)) \subseteq f(w, V(A))$.

iv) $f(w, V(A \vee B)) \subseteq f(w, V(A)) \bigcup f(w, V(B))$.

Notice that the following condition that intuitively represents (L) using selection functions is not satisfied by the function characterized by conditions (i)-(iv).

(L*) If $f(w, V(B)) \bigcap V(A) \neq \emptyset$, then $f(w, V(A)) \neq \emptyset$

To see this, consider the following definition of the selection function given by Delgrande in [9]:

Definition 2.6 Let $f(w, V(A)) = \{w_1 : w_1 \leq w, w_1 \models A$, and for all w_2 such that $w_2 \leq w_1$ and $w_2 \models A$, we also have $w_1 \leq w_2\}$.

Delgrande proved in [9] that if f is defined in this way, then it satisfies (i)-(iv). To construct a countermodel, take the relation $\leq$ as being a relation among rationals in the interval (0,1]. Let V(A) = (0, .9] and V(B) = {.5}. The antecedent of (L*) is satisfied but the consequent is not.

The same style of proof can be used to show non-equivalence of $\mathcal{V}$ and $\mathcal{NP}$.[2] Consider the theorem of $\mathcal{V}$: $(A > \neg A) \rightarrow (B > \neg A)$. The antecedent is satisfied because there are no closest A-worlds, yet the consequent is false because the only B-world is an A-world.

[2]To see a more complete account of the behaviour of (L*) in the richer setting of epistemic semantics of conditionals, and the relation of the condition with some of the axioms of V, see [1].

Furthermore, the above countermodel demonstrates the inappropriateness of translating $\Box A$ as $\neg A > A$ for systems like NP, whose models do not satisfy (L*). For example, in the above countermodel, $\neg A > A$ is satisfied, yet not all the worlds in the model are A-worlds. Hence, the (T) axiom will not be valid in some $\mathcal{NP}$ models, if the $\Box$ is defined in terms of the '>' in the standard way. Since the correct translation of the necessity operator in terms of the conditional for $\mathcal{NP}$-like systems is currently unknown, we will not deal syntactically with the universal and absolute extensions of $\mathcal{NP}$. This does not, however, preclude our using the universal and absolute models for $\mathcal{NP}$. Our use of these models will be crucial for the subsequent mappings.

The non-equivalence between $\mathcal{V}$ and $\mathcal{NP}$ also suggest that the system $\mathcal{NP}$ can be characterised in terms of sphere semantics by a pure set of sphere models (dropping the constraint (L) that characterize $\mathcal{V}$). In order to prove this conjecture we will show that the set of sphere models (without further constraint) is complete and sound with respect to $\mathcal{NP}$.

To give an alternative semantics, a constructive procedure will be followed: a function $f_\$$ will be constructed, defined from an order among worlds represented by a spheres model, and it will be proven this function is exactly the function f characterised by the axioms (i)-(iv) of Delgrande.

Definition 2.7 If $(\bigcup \$_w) \bigcap V(A) \neq \emptyset$ then let $f_\$(w, V(A)) = \bigcap\{S \in \$_w : S \bigcap V(A) \neq \emptyset\}$. Otherwise, let $f_\$(w, V(A)) = \emptyset$.

Theorem 2.1 *Let $\langle I, \$\rangle$ be any spheres model. If $f_\$(w, V(A))$ is defined for every world $w \in I$ and sentence $A \in L$ by Definition 2.7, then the axioms (i)-(iv) are satisfied.*

Theorem 2.2 *Let f be any function satisfying axioms (i)-(iv) of $\mathcal{NP}$. Then there is a spheres model, and a function $f_\$$ defined off of it, satisfying: $f(w, P(A)) = f_\$(w, V(A))$ for any world and sentence A of the language.*

Proof: Theorems 2.1 and 2.2 are proved in [2]. •

Theorem 2.2 states that for every $\mathcal{NP}$ selection function f, there exists a corresponding selection function $f_\$$ defined in relation to a spheres model. Hence, we are justified in using spheres models to represent the models of $\mathcal{NP}$ just in case the selection function used is the one set forth in Definition 2.7. Moreover, we are justified in using a truth definition for $\mathcal{V}$ models that employ selection functions. Hence, when dealing with $\mathcal{V}$ and $\mathcal{NP}$, we may use spheres models with a selection function truth definition. Notice further that Theorem 2.2 does not require that the equivalent spheres model be one that satisfies the limit assumption, hence confirming that the models of $\mathcal{NP}$ are not coincident with

those of $\mathcal{V}$.

2.4 Ordering Semantics

In the last section we developed a unified semantic account for the systems $\mathcal{NP}$ and $\mathcal{V}$ in terms of system of spheres semantics. This semantics will prove to be quite useful in the mappings with the nonmonotonic systems. Nevertheless the above system of spheres semantics does not provide a way of constructing a unified semantics for the $\mathcal{V}$, $\mathcal{NP}$ and $\mathcal{SS}^*$ family of conditional logics. Such a framework is essential if the various conditional systems are to be related one to another. Via the mapping in Section 4, we will then be able to show that the nonmonotonic systems form the same type of hierarchy as do the conditional logics.

Following Burgess in [8], we will present the unifying semantics using a ternary relation among worlds, instead of a system of spheres function \$.

Definition 2.8 An ordering model is a triple $\langle I, f, P\rangle$ where I is a non-empty set of worlds, R is a ternary relation on I, and P is a classical valuation whose extension for the conditional fragment of the language is given as follows:

> For $x \in I$, we set $I_x = \{y : \exists z R_{xyz} \vee R_{xzy}\}$. Then $P(A > B)$ is the set of all worlds $x \in I$ such that $\forall y \in (I_x \bigcap P(A))(\forall z \in (I_x \bigcap P(A))\neg Rxzy) \rightarrow y \in P(B)$.

We will now list a set of restrictions over the ordering models that will be useful in the following discussion:

(Tr) $\forall x \in I, \forall y, z, w \in I_x(R_{xyz} \wedge R_{xzw} \rightarrow R_{xyw})$

(N Tr) $\forall x \in I, \forall y, z, w \in I_x(\neg R_{xyz} \wedge \neg R_{xzw} \rightarrow \neg R_{xyw})$

(Irr) $\forall x \in I, \forall y \in I_x(\neg R_{xyy})$

(C) $\forall x \in I(x \in I_x \wedge \forall y \epsilon I_x(y \neq x \rightarrow R_{xxy})$

(T) $\forall x \in I(x \in I_x)$

(U) $\forall x, y \in I(I_x = I_y)$

(A) $\forall x, y \in I, \forall z, w \in I_y \forall z, w \in I_x(R_{xzw} \rightarrow R_{yzw})$

(L) $\forall x \in I \forall y \in M \exists z \in M \neg \exists r \in M(z \leq y \wedge r < z)$ where $M = (I_x \bigcap P(A))$

Theorem 2.3 (a) *The set of ordering models constrained by (Tr), (Irr) and (L) is sound and complete with respect to the system $\mathcal{SS}^*$.*
(b) *The set of ordering models constrained by (Tr), (Irr), (L) and (N Tr) is sound and complete with respect to the system $\mathcal{V}$.*
(c) *The set of ordering models constrained by (Tr), (Irr), and (N Tr) is sound and complete with respect to the system $\mathcal{NP}$.*

In the diagram below we present a hierarchy of systems. The diagram is intended to be interpreted as follows: Whenever one system is connected to another by a path of upward lines, the higher one is an extension of the other. The basic systems are $\mathcal{V}$, $\mathcal{NP}$ and $\mathcal{SS}^*$. The path from $\mathcal{SS}^*$ to $\mathcal{V}$ symbolises the addition of negative transitivity as a constraint on the ordering relation. The path from $\mathcal{NP}$ to $\mathcal{V}$ is effected by the addition of (L). The system $\mathcal{SS}^*$' included in the diagram is no immediate interest but has been included for the sake of symmetry. The (T), (TU) and (TA) extensions of $\mathcal{V}$, $\mathcal{NP}$ and $\mathcal{SS}^*$ are also represented.

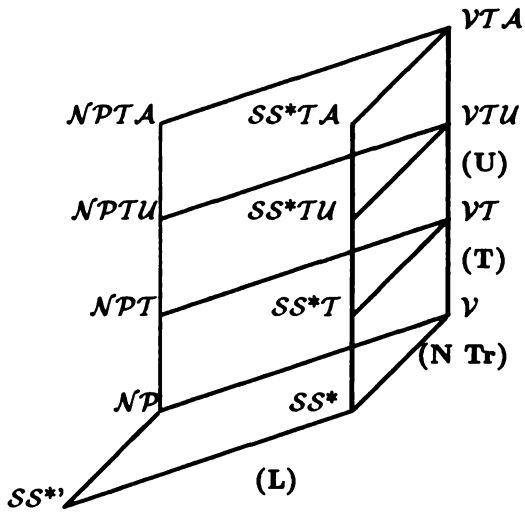


Figure I.

3 PREFERENTIAL AND RATIONAL LOGICS

In [13], Kraus, et al., present the following definition of a preferential model:

Definition 3.1 A preferential model W is a triple $\langle S, l, <\rangle$ where S is a set, the elements of which will be called states, $l : S \rightarrow U$ is a labelling function which assigns a world from the universe of reference to each state and $<$ is a strict partial order on S (i.e., an irreflexive, transitive relation) satisfying the following smoothness condition: for all a belonging to the underlying propositional language L, the set of states: $\hat{a} = \{s : s \in S, s \models a\}$ is smooth.

The smoothness condition can be defined in the following manner:

Definition 3.2 If $P \subseteq S$ and $<$ is a binary relation on S, P is a smooth subset of S iff $\forall t \in P$, either there exists an s minimal in P such that $s < t$ or t is itself minimal in P.

The symbol '$\models$' is characterised by the following defi-

nition:

Definition 3.3 In a given nonmonotonic model $\langle S, l, < \rangle$, for a state s and a formula a we write $s \Vdash a$ and read s satisfies a iff $l(s) \models a$, where '$\models$' is the classical notion of logical consequence.

The definitions introduced above allow for a modification of the classical notions of entailment and truth compatible with the idea of nonmonotonicity. The following definition shows how this task can be done:

Definition 3.4 Suppose a model $W = \langle S, l, < \rangle$ and $a, b \in L$ are given. The entailment relation defined by W will be denoted by $\mid\sim_W$ and is defined by: $a \mid\sim_W b$ iff for all s minimal in $\hat{a}$, $s \Vdash b$.

Note: $a \not\mid\sim_W b$ iff there exists an s minimal in $\hat{a}$, $s \not\Vdash b$.

Kraus, et al., then introduce the notion of preferential entailment, which is indistinguishable from the classical, Tarskian conception of logical consequence.

Definition 3.5 Let K be a set of conditional assertions. We say that K preferentially implies (P-implies) the conditional assertion $a \mid\sim b$ iff $a \mid\sim_V b$ for all preferential models V for which $\mid\sim_V$ contains all of the assertions of K.

The preferential models were used in [13] to define a family of preferential logics. In [13] it was also claimed that all reasonable nonmonotonic logics could be adequately described as preferential entailment relations. Nevertheless in [15], Lehmann and Magidor focused on a subfamily of preferential models – the ranked models.

Definition 3.6 A ranked model W is a preferential model $\langle S, l, < \rangle$ where the strict partial order $<$ may be defined in the following way: there is a totally ordered set W (the strict order on W will be denoted by $\angle$) and a function $r : S \to W$ such that $s < t$ iff $r(s) \angle r(t)$.

The effect of the function r is to rank the states, i.e. a state of smaller rank is more normal than a state of higher rank. Lehmann, et al., also introduce ranked models where the ordering of the states does not obey the smoothness requirement.

Observation 3.1 *A strict partial order $<$ is ranked iff it is negatively transitive, i.e., if $\neg(x < y)$ and $\neg(y < z)$ then $\neg(x < z)$.*

Definition 3.7 A rough ranked model V is a preferential model $\langle S, l, < \rangle$ for which the strict partial order $<$ is ranked and the smoothness requirement is dropped.

The entailment relations induced by ranked and rough ranked models can be defined in the same form as in Definition 3.4 and the notion of rational entailment can be characterized in the same way that preferential entailment was characterized in Definition 3.5.

From the syntactical point of view, Kraus, et al., proved a representation theorem for the following system $\mathcal{P}$ in terms of the above preferential models.

$$\text{(R)} \quad A \mid\sim A \qquad \text{(LLE)} \frac{\models A \leftrightarrow B,\ A \mid\sim C}{B \mid\sim C}$$

$$\text{(RW)} \frac{\models A \to B,\ C \mid\sim A}{C \mid\sim B} \qquad \text{(CM)} \frac{A \mid\sim B,\ A \mid\sim C}{A \wedge B \mid\sim C}$$

$$\text{(AND)} \frac{A \mid\sim B,\ A \mid\sim C}{A \mid\sim B \wedge C} \qquad \text{(OR)} \frac{A \mid\sim C,\ B \mid\sim C}{A \vee B \mid\sim C}$$

Lehmann, et al., prove that the system $\mathcal{R}$, complete with respect to ranked models, can be obtained adding the following rule of rational monotony to the above set of rules.

$$\text{(RM)} \frac{A \mid\sim C,\ \neg(A \mid\sim \neg B)}{A \wedge B \mid\sim C}$$

Finally, Lehmann, et al., in [15] suggested that the syntactic system $\mathcal{RR}$ obtained from $\mathcal{R}$ by dropping the rule (CM) is sound and complete with respect to rough ranked models. They obtain this conjecture from the work of James Delgrande in conditional logic. We will have the occasion to prove this conjecture true.

The surface similarities between conditional logic and the nonmonotonic logic of Kraus, et al., and Lehmann, et al., should now be immediate. Semantically, both conditional and nonmonotonic models involve strict partial orders over the domains of the models. Satisfaction of conditional assertions such as $a \mid\sim b$ or $a > b$ involve the determination of whether the minimal elements of the domain that satisfy a also satisfy b. The syntactic resemblences would lead one to conjecture that (the HF or GHF fragment of) many conditional systems coincide with the theses of $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{RR}$.

In the following sections we will show that these surface similarities are not deceiving, and that deep equivalences can be forged on both the semantic and syntactic level.

4 MAPPINGS

In the literature of nonmonotonic logic, the trend has been to use sets of possible worlds, instead of states, when dealing with preferential and rational models. See, e.g., Shoham[19], Boutilier[7], and Katsuno-Satoh[12]. Kraus, et al., have argued that their states

cannot be equated with maximal sets of propositions. They point out that whenever a model contains two states which bear the same label (i.e, the labelling function maps the two states to the same world), there does not exist an equivalent model with the same domain in which the labelling function is the identity function.

In the following section, we will show that while states are distinguishable from worlds, it is always possible to find, for rough ranked models in which the domain is a set of states, an equivalent rough ranked model in which the domain consists of possible worlds. This result will then be used to show that the states are eliminable for preferential models as well.

The states are eliminated using a two step process. First, it is shown that for each rough ranked model with states using the binary relation $<$, there exists one and only one equivalent spheres-like model with states. Second, these 'generalised' spheres models will be mapped to equivalent spheres models which contain only worlds. The restricted spheres models will be shown to be $\mathcal{NPTA}$ models.

In order to accomplish these mappings, we first introduce the following taxonomy of spheres models:

Definition 4.1 S is a nested set of sets iff $\forall U, V \in S, U \subseteq V$ or $V \subseteq U$.

Definition 4.2 Let a Generalised-spheres model [GS-model] be a triple $\langle D, f, l\rangle$, where D is a non-empty set, f is a function $f : D \rightarrow 2^{2^D}$, where $\forall S \in ran(f), S$ is nested, and l is a labelling function $l : D \rightarrow U$ (where U is the universe of reference).

Definition 4.3 Let a Generalised-absolute-spheres model [GAS-model] be a GS-model, where f is a constant function.

Definition 4.4 Let a Generalised-universal-absolute-spheres model [GTAS-model] be a GSA-model, where $\bigcup ran(f) = D$.

Definition 4.5 Let a Restricted-spheres model [RS-model] be a GS-model, where D is a set of possible worlds and l is the identity function.

We can then extend the taxonomy of GS-models for RS-models by defining RSA, RTAS models as we did above (definitions omitted).

We can show that for every non-trivial rough ranked model (i.e., a rough ranked model whose domain is non-empty), there exists one and only one Generalised-universal-absolute-spheres model, and vice versa.

Let $\mathbf{RR}^-$ be the class of all non-trivial rough ranked models and **GTAS** be the class of all GTAS-models. To prove that there exists a one-to-one correspondence between $\mathbf{RR}^-$ and **GTAS**, it is necessary to produce a bijective mapping from $\mathbf{RR}^-$ to **GTAS**. We first propose the following two definitions:

Definition 4.6 Let $\langle D, <, l\rangle$ be an rough ranked model. Then f' is a function $f' : D \rightarrow 2^{2^D}$, such that $\forall d \in D, f'(d) = \{S_x : x \in D\} \bigcup D$, where $S_x = \{y \in D : y < x\}$.

Definition 4.7 Let $\langle D, f, l\rangle$ be an GTAS-model. Then $d_1 <' d_2$ iff $\exists V \in f(d)(d_1 \in V \wedge d_2 \notin V)$, for some $d \in D$.

Definition 4.6 defines a function that maps every state in the domain of a non-trivial rough ranked model to one nested set of sets. The order of inclusion in the nested set is induced by the ranked relation of the rough ranked model. Definition 4.7 defines a binary relation between states by using the ordering of states induced by the nestedness of the system of spheres of a GTAS model.

Lemma 4.1 *Let D be any non-empty set, f' defined as in Definition 4.6, and l be any labelling function. $\langle D, f', l\rangle$ is a GTAS-model.*

Lemma 4.2 *Let D be any non-empty set, $<'$ defined as in Definition 4.7, and l be any labelling function. $\langle D, <', l\rangle$ is a non-trivial rough ranked model.*

We now propose the two main mapping functions between $\mathbf{RR}^-$ and **GTAS**:

Definition 4.8 Let g be the function g : **GTAS** $\rightarrow$ $\mathbf{RR}^-$ such that $g(\langle D, f, l\rangle) = \langle D, <', l\rangle$, where $<'$ is defined as in Definition 4.7.

Definition 4.9 Let h be the function h : $\mathbf{RR}^-$ $\rightarrow$ **GTAS** such that $h(\langle D, <, l\rangle) = \langle D, f', l\rangle$, where f' is defined as in Definition 4.6.

Lemma 4.3 *g is the inverse function of h.*

Proof: It must be proven that that $g(h(\langle D, <, l\rangle)) = \langle D, <, l\rangle$. Since D and l are left unchanged by definitions 4.8 and 4.9, we need only show that if $g(h(\langle D, <, l\rangle)) = \langle D, <', l\rangle$ then $<=<'$. (1) $<\subseteq<'$: Assume, for contradition, that for some $x, y \in D, \neg(x <' y)$ and $x < y$. By Definition 4.7, $\forall S(x \in S \rightarrow y \in S)$. Two cases are possible - a) For some $k, x \in S_k$. Hence, for any arbitrary k, if $x \in S_k$, then $y \in S_k$. By Definition 4.6, it follows that if $\neg(y < k)$ then $\neg(x < k)$. Consider the case of $k = y$. Since $\neg(y < y)$, by irreflexiveness, $\neg(x < y)$, contradicting $x < y$. b) For all $k, x \notin S_k$ but $x \in D$. Hence, by Definition 4.6, $\forall z \in D, z < x$ or $(\neg(z < x) \wedge \neg(x < z))$. Consider the case of $z = y$. This contradicts $x < y$. (2) $<'\subseteq<$: Assume, for contradition, that for some $x, y \in D, \neg(x < y)$ and $x <' y$. By Definition 4.7, there exists an S_k such that $x \in S_k$ and $y \notin S_k$. By Definition 4.6, $x < k$ and $\neg(y < k)$. Since $(x < k \wedge \neg(y < k) \rightarrow x < y)$, we conclude $x < y$. Contradiction. •

Lemma 4.4 *h is the inverse function of g.*

Proof: We must prove that $h(g(\langle D, f, l\rangle)) = \langle D, f, l\rangle$. Since D and l are left unchanged by definitions 4.8 and 4.9, one need only show that if $h(g(\langle D, f, l\rangle)) = \langle D, f', l\rangle$ then $\forall x \in D, f(x) = f'(x)$. We first assume $h(g(\langle D, f, l\rangle)) = \langle D, f', l\rangle$ and by the previous theorem we know that $g(\langle D, f', l\rangle) = \langle D, <', l\rangle$. Therefore, we now know that the models containing f and f', respectively, can be mapped to the same rough ranked model which contains $<'$. Hence, by Definition 4.7, $\forall x, y \in D(x <' y \leftrightarrow \exists S \in f(x)(x \in S \wedge y \notin S))$ and $\forall x, y \in D(x <' y \leftrightarrow \exists S \in f'(x)(x \in S \wedge y \notin S))$. By pure logic, $\forall x, y \in D((\exists S \in f(x)(x \in S \wedge y \notin S)) \leftrightarrow (\exists S \in f'(x)(x \in S \wedge y \notin S)))$. It is immediate, however, that this condition holds if and only if $\forall x \in D, f(x) = f'(x)$ (since f and f' are constant functions and map to only one nested set). •

Theorem 4.1 **RR$^-$** *and* **GTAS** *can be correlated one-to-one.*

Proof: By lemmas 4.3 and 4.4, both maps g and h are one-to-one and surjective by the existence of the corresponding inverse functions. Hence, g and h are bijective functions. •

We now show that the mappings introduced above are truth-preserving, i.e., if $a \mid\!\sim b$ is satisfied in a model in the domain of g or h, then the $a \mid\!\sim b$ will be satisfied in the image of those functions. In order to accomplish this, we first show that minimality is preserved by the mapping.

Definition 4.10 For any GS-model, X, and for any proposition $a \in L$, the a-minimum elements of $f(d)$ in X, for some $d \in D$, is defined as follows: $\tilde{a}^{GS} = //a// \bigcap (\bigcap \{S \in f(d) : S \bigcap //a// \neq \emptyset\})$, if $\bigcup (f(d) \bigcap //a//) \neq \emptyset$, where $//a// = \{d \in D : l(d) \models a\}$.

Definition 4.11 For any rough ranked model, RR, and for any proposition $a \in L$, the a-minimal elements of RR are defined as follows: $\tilde{a}^{RR} = \{d \in D : d \in //a//$ and $\forall e \in //a//, \neg(e < d)\}$.

Theorem 4.2 *Let X be a GTAS-model and $Y = g(X)$. $\forall d \in D, \forall a \in L$, d is an a-minimum element of $f(d)$ in X iff d is an a-minimal element of $<$ in Y.*

Proof: First assume by contradiction that $x \in \tilde{a}^{RR}$, but $x \notin \tilde{a}^{GS}$. Then $x \notin //a// \bigcap (\bigcap \{S \in f(d) : S \bigcap //a// \neq \emptyset\})$. Thus, there exists a T such that $T \bigcap //a// \neq \emptyset$, and $x \notin T$. Let $y \in T \bigcap //a//$. By Definition 4.7, $y < x$. But since $x \in \{d \in D : d \in //a//$ and $\forall e \in //a//, \neg(e < d)\}$, $\neg(y < x)$.

Second assume by contradiction that $x \notin \tilde{a}^{RR}$, but $x \in \tilde{a}^{GS}$. Then there exists an element y such that $y \in //a//$, and $y < x$. By Definition 4.7, there exists an $S \in f(d)$, such that $y \in S$ and $x \notin S$. In addition, $S \bigcap //a// \neq \emptyset$. By assumption, $x \in \tilde{a}^{GS}$, i.e., x is a member of all $S' \in f(d)$, such that $S' \bigcap //a// \neq \emptyset$. But as we showed above, there exists an $S \in f(d)$, where $S \bigcap //a// \neq \emptyset$ and $x \notin S$. •

Theorem 4.3 *Let Y be a non-trivial rough ranked model and $X = h(Y)$. $\forall d \in D, \forall a \in L$, d is an a-minimal element of $<$ in X iff d is an a-minimum element of $f(d)$ in Y.*

Proof: Similar to above proof. •

Definition 4.12 $a \mid\!\approx_S b$ where S is a GTAS-model iff $//a// \bigcap (\bigcap \{S \in f(d) : S \bigcap //a// \neq \emptyset\}) \subseteq //b//$.

The sign '$\mid\!\approx_S$' has been introduced because, stricly speaking, '$\mid\!\sim_S$' was defined only for models with a binary relation and not for spheres models, and the satisfaction of the operator '$>$' was defined for models that contain possible worlds, not states.

Notice that by Theorem 4.2 it follows trivially that if X is a GTAS-model and $Y = g(X)$, then $a \mid\!\approx_X b$ iff $a \mid\!\sim_Y b$. The biconditional holds as well if the h function is used to map the non-trivial rough ranked models to the GTAS models. Hence both g and h are truth-preserving maps. Since the maps are truth-preserving, they must also preserve validity of conditional assertions. Moreover, as the next theorem shows, the maps can be used to show the preservation of GHF patterns between GTAS and the class of all rough ranked models.

Definition 4.13 $a_1 \mid\!\approx b_1, \ldots, a_i \not\mid\!\approx b_i, \ldots, a_n \mid\!\approx b_n$ GTAS-implies $a \mid\!\approx b$ iff for all GTAS-models S where $a_1 \mid\!\approx_S b_1, \ldots, a_i \not\mid\!\approx_S b_i, \ldots, a_n \mid\!\approx_S b_n$, it is also the case that $a \mid\!\approx_S b$.

Theorem 4.4
$a_1 \mid\!\approx b_1, \ldots, a_i \not\mid\!\approx b_i, \ldots, a_n \mid\!\approx b_n$ GTAS-implies $a \mid\!\approx b$ iff $a_1 \mid\!\sim b_1, \ldots, a_i \not\mid\!\sim b_i, \ldots, a_n \mid\!\sim b_n$ RR-implies $a \mid\!\sim b$.

Proof: Immediate for non-trivial rough ranked models based on the one-to-one correspondence between GTAS and RR$^-$. We need only show that every GHF pattern valid in GTAS is also satisfied in the trivial rough ranked model. This claim is immediate, however, because the only way that a GHF pattern can fail is if the right-hand side of the pattern is not satisfied. But all conditional assertions are satisfied in the trivial model. •

We now show that for every Generalised-universal-absolute-spheres model, there exists an equivalent Restricted-universal-absolute-spheres model, i.e., a spheres model that is universal and absolute but which consists only of possible worlds.

Definition 4.14 Let $\langle D, f, l\rangle$ be an GTAS-model. Then $W = \{w : \exists s \in D, l(s) = w\}$.

Definition 4.15 Let $\langle D, f, l\rangle$ be an GTAS-model. Then f^* is a function $f^* : W \rightarrow 2^{2^W}$, such that for

some $d \in D$ and $\forall w \in W, f^*(w) = \{T_S : S \in f(d)\}$, where $T_S = \{v : \exists s \in S$ and $v = l(s)\}$ and W is defined as in Definition 4.14.

In order to construct an equivalent RTAS model from a GTAS model, we take as the domain W the set of labels of the states in the GTAS model, as is done in Definition 4.14. The function f^* of Definition 4.15 is the constant function that maps from each world in the domain to a set which contains, for every sphere of states in the image of the f function of the GTAS model, a corresponding sphere composed of the labels of those states.

Lemma 4.5 *Let W be a set defined as in Definition 4.14, f^* defined as in Definition 4.15, and l be any labelling function. $\langle W, f^*, l\rangle$ is a RTAS-model.*

Let **RTAS** be the class of all RTAS-models. We now propose the following mapping function from **GTAS** to **RTAS** (the map from **RTAS** to **GTAS** is trivial because **RTAS** is properly contained in **GTAS**).

Definition 4.16 Let p be the function p : **GTAS** $\rightarrow$ **RTAS** such that $p(\langle D, f, l\rangle) = \langle W, f^*, l\rangle$, where W is defined as in Definition 4.14, and f^* is defined as in Definition 4.15.

As we did before, we now show that the mapping function p is truth-preserving.

Definition 4.17 For any RS-model Y, and for any proposition $a \in L$, the a-minimum elements of $f(w)$ in Y, for some $w \in W$, is defined as follows: $\tilde{a}^{RS} = /a/ \bigcap (\bigcap \{S \in f(w) : S \bigcap /a/ \neq \emptyset\})$, if $\bigcup (f(w) \bigcap /a/) \neq \emptyset$, where $/a/ = \{w \in W : l(w) \models a\}$.

Lemma 4.6 $\forall d \in D$, *if* $d \in \tilde{a}^{GS}$ *then* $l(d) \in \tilde{a}^{RS}$.

Lemma 4.7 $\forall w \in W$, *if* $w \in \tilde{a}^{RS}$ *then* $\exists s \in D, s \in \tilde{a}^{GS}$, *where* $l(s) = w$.

Theorem 4.5 *Let X be a GTAS-model and $p(X) = \langle W, f^*, l\rangle$. Then* $\forall a, b \in L, \tilde{a}^{GS} \subseteq //b//$ *iff* $\tilde{a}^{RS} \subseteq /b/$.

Proof: In order to prove the 'only if' side of Theorem 4.5 we assume for contradiction that $\tilde{a}^{GS} \subseteq //b//$, and for some $w \in \tilde{a}^{RS}$ and $w \notin /b/$. By Lemma 4.7, if $w \in \tilde{a}^{RS}$ then $\exists s \in D, s \in \tilde{a}^{GS}$, where $l(s) = w$. Since there exists a state s which is a member of $\tilde{a}^{GS}$, s is a b-state by the initial assumption. Hence, $l(s)$ is a b-world. Contradiction.

The 'if' part is proved in a similar manner. We assume for contradiction that $\tilde{a}^{RS} \subseteq /b/$, and for some state $d \in \tilde{a}^{GS}$, $d \notin //b//$. By Lemma 4.6, if $d \in \tilde{a}^{GS}$ then $l(d) \in \tilde{a}^{RS}$. By assumption, $l(d)$ is a b-world. Hence, $d \in //b//$. Contradiction. •

By Theorem 4.5, if X is a GTAS-model and $Z = p(X)$, then $a \mid\!\sim_X b$ iff $\models_Z a > b$. Thus, the mapping (via the p function) from GTAS models to RTAS is truth preserving, and *a fortiori*, validity preserving. However, even though the mapping between GTAS and RTAS is truth-preserving, it is obviously not one-to-one. Hence, the preservation of GHF theorems through the mapping is not as immediate, although the proof is simple.

Definition 4.18 $a_1 > b_1, \ldots, \neg(a_i > b_i), \ldots, a_n > b_n$ RTAS-implies $a > b$ iff for all RTAS-models T where $\models_T a_1 > b_1, \ldots, \not\models_T a_i > b_i, \ldots, \models_T a_n > b_n$, it is also the case that $\models_T a > b$.

Theorem 4.6
$a_1 \mid\!\sim b_1, \ldots, a_i \not\mid\!\sim b_i, \ldots, a_n \mid\!\sim b_n$ *GTAS-implies* $a \mid\!\sim b$ *iff* $a_1 > b_1, \ldots, \neg(a_i > b_i), \ldots, a_n > b_n$ *RTAS-implies* $a > b$.

Proof: Immediate in the 'only if' direction because all RTAS-models are GTAS-models. In the 'if' direction, we assume for contradiction that $a_1 > b_1, \ldots, \neg(a_i > b_i), \ldots, a_n > b_n$ RTAS-implies $a > b$ but for some GTAS-model $S, a_1 \mid\!\sim_S b_1, \ldots, a_i \not\mid\!\sim_S b_i, \ldots, a_n \mid\!\sim_S b_n$ and $a \not\mid\!\sim_S b$. By Theorem 4.5, there exists an equivalent RTAS-model T such that $\models_T a_1 > b_1, \ldots, \not\models_T a_i > b_i, \ldots, \models_T a_n > b_n$ and $\not\models_T a > b$. Contradiction. •

5 Minimal Logics

In the previous section, it was proven that every rough ranked model with states is equivalent to an RTAS model composed solely with possible worlds. Based on the discussion of conditional logic in Section 2, it is clear that these RTAS models are the absolute and universal extensions of Delgrande's $\mathcal{NP}$ models, since both types of models do not obey the Limit Assumption (see Theorem 2.2). What we have shown, therefore, is that the universal and absolute extensions of $\mathcal{NP}$ models validate the same GHF patterns as RR models do.

This result provides two benefits. First, since minimality is preserved from rough ranked models to GTAS models (See Theorem 4.3) and from GTAS models to RTAS models (See Lemmas 4.6 and 4.7), if a rough ranked model is stoppered (i.e., obeys the smoothness constraint), then the equivalent RTAS model identified by the mapping with also be stoppered (and vice versa). Based on the discussion in Section 2, we can identify stoppered RTAS models as the $\mathcal{VTA}$ models of Lewis. Moreover, since by Theorem 2.2, selection function semantics and the Lewis truth definitions are equivalent, we know that the theorems of the GHF counterpart of $\mathcal{VTA}$ are coincident with the theses of the system $\mathcal{R}$.

Second, as the next theorem shows, RTAS models can be mapped to RS models, with respect to GHF patterns. This will allow us to show that $\mathcal{NP}$ and $\mathcal{V}$ are the minimal classical conditional logics whose GHF

counterparts correspond to $\mathcal{RR}$ and $\mathcal{R}$.

Definition 5.1 $a_1 > b_1, \ldots, \neg(a_i > b_i), \ldots, a_n > b_n$ RS-implies $a > b$ iff for all RS-models U where $\models_U a_1 > b_1, \ldots, \not\models_U a_i > b_i, \ldots, \models_U a_n > b_n$, it is also the case that $\models_U a > b$.

Theorem 5.1 *$a_1 > b_1, \ldots, \neg(a_i > b_i), \ldots, a_n > b_n$ RS-implies $a > b$ iff $a_1 > b_1, \ldots, \neg(a_i > b_i), \ldots, a_n > b_n$ RTAS-implies $a > b$.*

Proof: Immediate in the 'only if' direction because all RTAS-models are RS-models. In the 'if' direction, we assume for contradiction that $a_1 > b_1, \ldots, \neg(a_i > b_i), \ldots, a_n > b_n$ RTAS-implies $a > b$ but for some RS-model $U = \langle W, f, l \rangle$, $\models_U a_1 > b_1, \ldots, \not\models_U a_i > b_i, \ldots, \models_U a_n > b_n$ and $\not\models_U a > b$. Hence, there exists some world of reference $w \in W$, such that in $f(w), \tilde{a}_1^{RS} \subseteq /b_1/, \ldots, \tilde{a}_i^{RS} \not\subseteq /b_i/, \ldots, \tilde{a}_n^{RS} \subseteq /b_n/$ and $\tilde{a}^{RS} \not\subseteq /b/$. Consider the model $T = \langle W, f^\circ, l \rangle$, where f° is defined as: $\forall v \in W, f^\circ(v) = f(w) \bigcup W$. T is therefore a RTAS-model. In addition, it is the case that in $f^\circ(v)$, for any $V \in W, \tilde{a}_1^{RS} \subseteq /b_1/, \ldots, \tilde{a}_i^{RS} \not\subseteq /b_i/, \ldots, \tilde{a}_n^{RS} \subseteq /b_n/$ and $\tilde{a}^{RS} \not\subseteq /b/$. This contradicts the assumption that $a_1 > b_1, \ldots, \neg(a_i > b_i), \ldots, a_n > b_n$ RTAS-implies $a > b$. •

According to Theorem 5.1, RTAS and RS models are equivalent with respect to GHF patterns. By similar arguments used at the beginning of this section, it follows that the theses of $\mathcal{R}$ and $\mathcal{RR}$ are coincident with the theorems of the GHF counterparts of $\mathcal{V}$ and $\mathcal{NP}$. This theorem also proves that the universal and absolute extensions of $\mathcal{V}$ and $\mathcal{NP}$ are non-creative with respect to GHF theses.

The result of the above mappings can be extended, showing that $a_1 > b_1, \ldots, a_n > b_n$ RS-implies $\neg(a > b)$ if and only if $a_1 \mathrel{|\!\sim} b_1, \ldots, a_n \mathrel{|\!\sim} b_n$ RR-implies $a \not\mathrel{|\!\sim} b$. This holds trivially—consider the trivial rough ranked model and a conditional model with an empty system of spheres.

These mappings also allow us to find the minimal conditional logics for the preferential system $\mathcal{P}$. We accomplish this task by showing that the HF theses of $\mathcal{P}$ (i.e., GHF theses which do not allow for negated snakes on the left side of the 'P-implies') are coincident with the theses of the HF counterpart of $\mathcal{SS}^*\mathcal{TA}$ (i.e., the universal and absolute extensions of $\mathcal{SS}^*$).

Definition 5.2 $a_1 > b_1, \ldots, a_n > b_n$ $\mathcal{SS}^*\mathcal{TA}$-implies $a > b$ iff for all $\mathcal{SS}^*\mathcal{TA}$-models V where $\models_V a_1 > b_1, \ldots, \models_V a_n > b_n$, it is also the case that $\models_V a > b$.

Theorem 5.2 *$a_1 \mathrel{|\!\sim} b_1, \ldots, a_n \mathrel{|\!\sim} b_n$ P-implies $a \mathrel{|\!\sim} b$ iff $a_1 > b_1, \ldots, a_n > b_n$ $\mathcal{SS}^*\mathcal{TA}$-implies $a > b$.*

Proof: Immediate in the 'only if' direction because all $\mathcal{SS}^*\mathcal{TA}$-models are $\mathcal{P}$-models. This follows because for $\mathcal{SS}^*\mathcal{TA}$ models, the ternary relation is inessential and the models are equivalent to models with a binary relation (i.e., a $\mathcal{P}$-model consisting only of worlds). In the 'if' direction, we assume for contradiction that $a_1 > b_1, \ldots, a_n > b_n$ $\mathcal{SS}^*\mathcal{TA}$-implies $a > b$ but that for some $\mathcal{P}$-model P $a_1 \mathrel{|\!\sim}_P b_1, \ldots, a_n \mathrel{|\!\sim}_P b_n$, and $a \not\mathrel{|\!\sim}_P b$. By Lemma 18 of Lehmann, et al. [13], for every $\mathcal{P}$-model P where $a_1 \mathrel{|\!\sim}_P b_1, \ldots, a_n \mathrel{|\!\sim}_P b_n$, and $a \not\mathrel{|\!\sim}_P b$, there exists an $\mathcal{R}$-model R where $a_1 \mathrel{|\!\sim}_R b_1, \ldots, a_n \mathrel{|\!\sim}_R b_n$, and $a \not\mathrel{|\!\sim}_R b$. By Theorem 4.5, this $\mathcal{R}$-model R with states (GTAS-model) is equivalent to some $\mathcal{VTA}$-model V with worlds (RTAS-model). Because all $\mathcal{VTA}$ models are GTAS models, there exists a $\mathcal{R}$-model R', by Theorem 4.2 and 4.3, which is equivalent to that $\mathcal{VTA}$ model V. This $\mathcal{R}$-model R' (which might be different from R) will consist of a domain of states where the states can be identified with worlds. We also know that this $\mathcal{R}$-model R' is also an $\mathcal{SS}^*\mathcal{TA}$ model. Hence, there exists some $\mathcal{SS}^*\mathcal{TA}$ model where $\models_V a_1 > b_1, \ldots, \models_V a_n > b_n$, and $\not\models_V a > b$. Contradiction. •

A consequence of this theorem is that states are completely eliminable for preferential models as well. The mapping between $\mathcal{R}$ and $\mathcal{RR}$ allowed us to derive this conclusion without finding the equivalent models that contain only worlds. It follows that, in order to derive the HF theorems of $\mathcal{P}$, it is only necessary to use the $\mathcal{SS}^*\mathcal{TA}$ models.

Nevertheless, $\mathcal{SS}^*\mathcal{TA}$ is not the minimal logic in the $\mathcal{SS}$ family of conditional systems. It can easily be shown that $\mathcal{SS}^*$ is the minimal logic that corresponds to $\mathcal{P}$ in the same manner that $\mathcal{V}$ and $\mathcal{NP}$ were shown to be the minimal logics for $\mathcal{R}$ and $\mathcal{RR}$. One needs only to show that the HF theorems of $\mathcal{SS}^*\mathcal{TA}$ are coincident with those of $\mathcal{SS}^*$. In one direction, the proof is immediate because all $\mathcal{SS}^*\mathcal{TA}$-models are $\mathcal{SS}^*$-models. In the other direction, the proof is again by contradiction. One assumes that a HF pattern is a theorem of $\mathcal{SS}^*\mathcal{TA}$ but not of $\mathcal{SS}^*$. If the HF pattern is not a theorem of $\mathcal{SS}^*$, then for one model and for one world of reference w in the domain W of that model, the ternary relation R induces an ordering among worlds such that the set K of conditional assertions and the conditional assertion $(a > b)$ do not fall in the relation '$\mathcal{SS}^*$-implies.' But then it is possible to recover a binary relation from the ternary relation by ignoring the first position that w occupies, i.e., $\forall x, y \in W, R'xy \leftrightarrow Rwxy$. This new binary relation R', coupled with the labelling function and the domain W, can be used to form an $\mathcal{SS}^*\mathcal{TA}$ model where the ordering among worlds will satisfy the conditional assertions of K but not $(a > b)$, which is a contradiction.

6 RELATED WORKS

We have showed that the (non-trivial) preferential models $\mathcal{R}$ and $\mathcal{RR}$ 'are' absolute and universal ex-

tensions of $\mathcal{V}$ and $\mathcal{NP}$ models, respectively. These structures, then, have the power of validating patterns which contain nested conditionals, although Kraus, et al., and Lehmann, et al., do not utilize this extra power. Nevertheless, the intuitive and formal characterization of nonmonotonic patterns of this type is still an open problem. Some conditions like 'defeasible centering' $(A > (B \rightarrow (A > B)))$ are not validated by RTAS models, and these models validate patterns whose nonmonotonic interpretation is quite unclear.

Some authors have observed that other models that are capable of validating specific nested axioms can also provide a complete characterization of T-extensions of $\mathcal{P}$ and $\mathcal{R}$ (in general, these authors did not analyze the problem of $\mathcal{RR}$). In fact, Craig Boutilier in [7] and Philippe Lamarre in [14] proved independently of each other that the flat fragment of a suitable conditional presentation of the modal system $\mathcal{S}4$ is capable of such a characterization. It is interesting to notice that the rewritten conditional version of $\mathcal{S}4$ ($\mathcal{C}4$ in Lamarre's terminology) can be expressed as our system $\mathcal{SS}^*$ enriched with additional nested axioms. According to these authors, the flat fragment of $\mathcal{C}4$ coincides with T-extension of $\mathcal{P}$; however, as we have shown, in order to get $\mathcal{P}$, only the models of $\mathcal{SS}^*$ are needed. It thus appears that the nested axioms only play the role of guaranteeing a complete coincidence of $\mathcal{C}4$ with $\mathcal{S}4$, but that they do not play any interesting role with respect to the characterization of $\mathcal{P}$. If the goal is to find a minimal conditional system coincident with $\mathcal{P}$ it is not necessary to use $\mathcal{C}4$ when you can directly use $\mathcal{SS}^*$. Nevertheless, the expression of the T-extensions of $\mathcal{P}$ and $\mathcal{R}$ as fragments of modal systems seems to be useful under a computational point of view, taking into account certain recent works in decision procedures of conditional logic (see for example [11]).

Katsuno and Satoh in [12] provided a three-way mapping, connecting belief revision operators, nonmonotonic consequence relations and the $>$'s of conditional logic. They provided a mapping only for stoppered consequence relations (thus they did not consider the system $\mathcal{RR}$ and its conditional counterpart). The conditional counterparts they selected for $\mathcal{R}$ and $\mathcal{P}$ are not minimal (they are stronger than $\mathcal{V}$ and $\mathcal{SS}^*$, respectively). In addition, they are not intuitive as systems for the defeasible conditional either. In the case of $\mathcal{R}$, the correspondent conditional system is Lewis' $\mathcal{VW}$ obtained from $\mathcal{V}$ by the addition of (MP). We already argued that modus ponens for '$>$' is unnatural when '$>$' is interpreted as defeasible implication. Nonetheless, it is true that the GHF counterparts of $\mathcal{VW}$ and $\mathcal{V}$ coincide (this surprising result is shown in [3]). It is not entirely clear whether Katsuno, et al., are using our GHF fragment as the conditional counterparts of $\mathcal{R}$ and $\mathcal{P}$. If they are, their result is formally correct, although not very compelling under an intuitive point of view.

John Bell analyzed the problem of the relation between nonmonotonic and conditional logic in two recent articles ([4] and [5]). In [4], he proposed Pollock's $\mathcal{SS}$ (called $\mathcal{C}$ by Bell) as the logic of preference logics. But it is clear that $\mathcal{SS}$ will not do. In fact, the pattern (M) $\neg(T > \neg A) \rightarrow (T > A)$ is derivable from the axiom (AS),[3] but it does not hold in any preferential system.

In proposition 5.2 of [5] Bell proposed some correspondences between conditional and nonmonotonic systems. Some of the correspondences established coincide with the ones proposed here, but others are incorrrect. First, Bell claims that $\mathcal{R}$ corresponds to the GHF counterpart of Lewis' $\mathcal{VC}$. But this is false, since the pattern (M) that belongs to GHF counterpart of $\mathcal{VC}$ does not hold in any preferential system. Secondly, Bell claims that $\mathcal{R}$ corresponds to the GHF counterpart of $\mathcal{NP}$,[4] but the pattern $(A > \neg A) \rightarrow (B > \neg A)$ does not have any correlate in the GHF counterpart of $\mathcal{NP}$. Finally, Bell argues that $\mathcal{P}$ corresponds to the GHF counterpart of $\mathcal{C}$, but, again, (M) belongs to the GHF counterpart of $\mathcal{C}$, and this pattern does not hold in $\mathcal{P}$.

7 CONCLUSION

We have shown that the systems of conditional logic that correspond to $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{RR}$ are deontic, which is to say non-centered, systems. Hence, the rules (MP) and (AS) are not valid inference schema for these logics. The fact that (MP) and (AS) are invalid in the nonmonotonic systems of Kraus, et al., and Lehmann, et al., cannot be ascribed, therefore, to the limitations of the language that these authors impose. While it is true that the inference patterns used in detaching the conditional cannot be represented in $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{RR}$, extending the language will not validate these inference patterns. The preferential models themselves are not centered and hence cannot be employed as semantic structures which sanction an actual 'jump to a conclusion.' The analogue in deontic conditional logic is clear – if a state of affairs B ought to be the case if A were to be the case, it is not necessarily true that B will be the case when A is the case.

From knowledge of A and $A \mid\sim B$, a reasoner who only has a preferential system at his disposal has no manner of extracting a conclusion of the type 'in the normal course of events B'. In addition, from the knowledge that A (or A holds in the normal case) and $A \mid\sim B$, she cannot conclude B is true either.

The parallels between the conditional logic approach and the one based on consequence relations now becomes clearer. Both approaches offer the possibility

[3] We owe this observation to David Makinson.

[4] This observation is probably based on the fact (see [6]) that Bell believes (wrongly as we proved before) that $\mathcal{V}$ and $\mathcal{NP}$ coincide.

of constructing an appropriate representational framework for defaults. In fact, the use of a non-classical preferential relation offers the possibility of representing the joint satisfaction of sequences of conditional assertions (with true antecedents) that obey the following pattern: as the antecedent is strengthened, the consequent flips to its negation. Significantly as well, both approaches do not contain mechanisms for drawing conclusions about the actual world from default rules. Thus, the deontic conditional and nonmonotonic consequence relation approaches can offer (1) an adequate representational tool and (2) an inferential apparatus appropriate for purely prototypical reasoning (from knowledge of the normality of A and $A \mid\sim B$, we can of course conclude the normality of B, for example). In addition, the conditional and preferential models used for representational purposes can be exploited in order to supplement these systems with appropriate inductive procedures. Whether the adequate pragmatics for such 'jumping' should employ the methods of statistics, belief revision theory, default logic, or a combination of all of these approaches remains an open problem for both the AI and philosophical communities.

ACKNOWLEDGEMENTS

An earlier version of this paper was presented at the University of Toronto, Computer Science Department. We would like to thank Carlos Alchourron, Peter Hilal, Isaac Levi and, especially, David Makinson for helpful comments on previous drafts.

References

[1] Arlo-Costa, H. "Conditionals and monotonic belief revisions: the success postulate" *Studia Logica* (1990).

[2] Arlo-Costa, H., Carnotta, R. "Non-monotonic Logics: Consequence Relations and Conditional Operators", *6th Simposio Brasileiro em Inteligencia Artificial* 328-43 (1989).

[3] Arlo-Costa, H. "Epistemic semantics for unnested conditionals and nonmonotonic notions of consequence" Submitted for publication (1992).

[4] Bell, J. "The logic of non-monotonicity" *Artificial Intelligence* 365-74 (1990).

[5] Bell, J. "Pragmatic Logics" *Proceedings KR'91* (1991).

[6] Bell, J. "Notes on Delgrande's normality conditional logic" Manuscript. computer Science Department, QMW, University of London, London E14NS, England.

[7] Boutilier, C. "Viewing Conditional Logics of Normality as Extensions of the Modal System S4" *Technical Report KRR-TR-90-4, Department of Computer Science, University of Toronto* (June 1990).

[8] Burgess, J. "Quick Completeness Proofs for Some Logics of Conditionals", *22 Notre Dame Journal of Formal Logic* 76-84 (1981)

[9] Delgrande, J. "A first-order logic for prototypical properties" *33 Artificial Intelligence* 105-130 (1987).

[10] Gabbay, D. "Theoretical foundations for nonmonotonical reasoning in expert systems" *Logics and Models of Concurrent Systems* (K.R. Apt., ed. 1985).

[11] Groeneboer, M. "Automated theorem proving in conditional logic" *Master thesis. Simon Frazer University* (1987).

[12] Katsuno, H., Satoh, K. "A Unified View of Consequence Relation, Belief Revision and Conditional Logic" *Proceedings IJCAI-91*

[13] Kraus, S., Lehmann, D., and Magidor, M. "Nonmonotonic reasoning, preferential models and cumulative logics" *44 Artificial Intelligence* 167-207 (1990).

[14] Lamarre, P., "S4 as the conditional logic of nonmonotonicity" *Proceedings KR'91* (1991).

[15] Lehmann, D., Magidor, M. (to appear) "Rational logics and their models; A study in cumulative logics." *Technical Report TR 88-16 of the Dept. of Computer Science, Hebrew University of Jerusalem* November 1988.

[16] Lewis, D. *Counterfactuals* (1973).

[17] Makinson, D. "General Theory of Cumulative Inference" *Non-Monotonic Reasoning* (M. Reinfrank, J. de Kleer, M. Ginsberg, and E. Sandewall eds., 1988).

[18] Pollock, J. "A refined theory of counterfactuals" *Journal of Philosophical Logic 10* 239-266 (1981).

[19] Shoham, Y. *Reasoning About Change* (1987).

On the Connection between Non-Monotonic Inference Systems and Conditional Logics

Gabriella Crocco
IRIT - Université Paul Sabatier
118, Route de Narbonne
31062 Toulouse Cedex - France
email: crocco@irit.fr

Philippe Lamarre
IRIT - Université Paul Sabatier
118, Route de Narbonne
31062 Toulouse Cedex - France
email: lamarre@irit.fr

Abstract

Recently some general frameworks have been proposed to classify and represent non-monotonic inference relations. In this paper, following on from these researches, a complete and general proof is given of the equivalence between some classes of non-monotonic inference relations and conditional logics. It is based on a systematic translation of expressions of one system into expressions of the other. Using the well-known conditional logics semantics, representation theorems follow without any difficulty. A comparison is made between conditional logic semantics and the semantics proposed by Katsuno, Satoh [Katsuno 91] and Bell [Bell 91]. Preferential systems are considered as a study case for the application of our general result.

1 INTRODUCTION

Since the concept of 'non-monotonic reasoning' was introduced in the computer science field, in the early seventies, many different non-monotonic logics have been developped (see [LeaSombe 89] for a survey of these formalisms).

Recently, D. Gabbay [Gabbay 85] suggested studying non-monotonic systems via the properties of the non-monotonic deduction relation they define[1]. This idea has been very fruitful. Many results have been obtained, especially with regard to the order and the classification of the different non-monotonic logics families. One of the main research lines has been to define "representation theorems" which associate semantics to non-monotonic deduction relations [Kraus 90] [Freund 91] [Lehmann 89] [Gärdenfors 91b]. Basically, these semantics represent the non-monotonic inference relations, i.e. a model corresponds to an inference relation (and vice versa). These theorems use technical devices that can be very different according to the properties which have been considered to define the inference relation. Even if, for some of them, simple and unified semantics were given (based on preferential relations over states as first proposed by Shoham [Shoham 88]), it was still a little disappointing to have so many different semantics (and so complicated) for simpler systems.

More recently, general semantical frameworks have been defined for some different non-monotonic systems [Nejdl 91] [Bell 91] [Katsuno 91]. Nejdl constructs a ternary relation. Katsuno and Satoh define structures, based on possible worlds, with pre-order and total pre-order over them (preferential relation or accessibility relation)[2]. This gives a model structure for preferential and rational[3] inference systems. This structure is a slight modification of sphere semantics defined by Lewis for counterfactuals [Lewis 86] (a total pre-order on possible worlds exactly defines a sphere system, and a pre-order may be seen as a generalization of it). Nevertheless as the model is based on an accessibility relation (as in [Kraus 90] [Lehmann 89] [Freund 91]), even if it was possible to describe systems which are less strong than cumulative ones, the needed structure would be much more complicated.

Bell defines a 'pragmatic canonical model' as a set I of interpretations, a meaning function and a selection function. The only difference with the models defined by Lewis in [Lewis 86] and extended to other conditional systems in Nute [Nute 80], is that the starting world (or the actual world) is not a parameter of the selection function. This is possible since Bell only considers formulae which are Boolean combinations of flat conditionals. Nested

[1] In fact, these relations differ from the monotonic one, not only because they do not satisfy monotonicity, (as the deduction relations of relevant logics or linear logic do) but also because they are incompatible with what is known as the cut rule in the logic field (i.e. the addition of the cut rule does not give a conservative extension of the system). The failure of this property has often incited logicians to deny the status of "logic" to these systems as it seems to imply the collapsing of all notions of derivation and a substantial opacity of the meaning of the connectives.

[2] This structure is extended to families of ordered structures: to each world is associated a family, just in a same way as in Lewis' semantics, a sphere system is asociated to each world.

[3] See [Kraus 90] for the definition of these non monotonic inference relations.

occurrences of the conditional operator, combinations of classical and conditional formulae are not allowed. Using this structure Bell sketches some correspondence theorems for some non-monotonic systems.

This paper follows on from these previous works. We give a complete and general proof of the equivalence between conditional logics and definitions of non-monotonic inference relations by their properties. Our proof is based on a translation of non-monotonic expressions in conditional ones. Using the well-known semantics of conditional logics, representation theorems follow without any difficulty.

The connection between non-monotonic inference relations and conditional logics has been discussed in [Kraus 90] and [Bell 91]. It was first noted by Van Benthem (as reported in [Shoham 87]) and improved in [Van Benthem 88]). It is also fundamental in [Bell 90] and [Delgrande 88]. Nevertheless, even if all these authors recognize that the conditional connective '$\Rightarrow$' encodes a non-monotonic deduction relation in the language, they do not use directly conditional logics to obtain a unifying semantical framework. The two main criticisms against considering a large class of non-monotonic inference systems as only a typographical variant of the corresponding well-known conditional logics are presented in [Kraus 90] and [Bell 91]. They can be summarized in the following terms:

a) Nested non-monotonic symbols do not have any meaning while conditionals can be nested.

b) The conditional semantics evaluation refers to the actual world while this does not make any sense within the framework of non-monotonic inferences.

These two problems are strictly linked. Indeed, if the language is restricted to Boolean combinations of flat conditionals, keeping or dropping the actual world index does not change anything. Among these two points, the second is the most restrictive one since it removes nested conditionals and also formulae of the form $(A\Rightarrow B)\rightarrow(A\rightarrow B)$. But let us gently polemize for a while.

It is true that from a syntactical point of view, $(A\mid\sim B)\mid\sim C$ is a strange construction, but no more than $(A\mid\!-B)\mid\!-C$, that is normally written $(A\rightarrow B)\rightarrow C$. The real problem is to give an acceptable intuitive meaning to this construction and we think that there is no conclusive reason to claim that there is none. For example, take the following reading "Considering that A is a sufficient ground to assert that B leads us to affirm C". The kind of information which could be extracted from these constructions, depends on the way nested conditionals are axiomatized, but this is another matter. Note that these remarks have also been introduced in the conditional logics literature. Lewis and Nute express some reserves for nesting conditionals since nested counterfactuals are not common in natural language. Delgrande agrees with them arguing that the meaning of nested conditionals is not clear [Delgrande 88]. So he restricts the language of his conditional logic NP to formulae without nested conditional operators. However, just consider the intuitive reading he gives of " $A\Rightarrow B$": "normally if A then B". The formula $(((A\Rightarrow B)\wedge A)\rightarrow B)$ is not NP-valid. Indeed, it is possible to know that "Normally if A then B" and also "A" but to be in such an exceptional case that B is not true. Considering the same hypothesis, and assuming that the considered case is as normal as possible, it would be intuitive to conclude that B is true. This can only be expressed using a nested conditional as in $(((A\Rightarrow B)\wedge A)\Rightarrow B)$. It is very interesting to note that this last formula is valid considering S4.3 (which allows nested operators) instead of NP. This latter system has been presented by Delgrande as the modal logic equivalent to NP[4], the accessibility relation translating some preference notions over possible worlds [Boutilier 90] [Lamarre 91].

Many different strategies have been proposed to avoid dealing with nested occurrences of conditionals: the language can be restricted; the semantics can be adapted, the index for the actual world can be erased; axioms can be added to analyze nested conditionals. However all these strategies may be done inside of the framework of conditional logics so there is no particular reason to abandon them.

The great advantage of conditional logics is that they have already been well studied, and they have clear syntactical metaproperties such as decidability, compactness and so on. Some of the results concerning conditional logics (such as minimal axiomatisation, inclusion relations between different systems, different possible semantics, correspondence with modal logics, ...) can be usefully applied to the study of non-monotonic properties.

The outline of this paper is the following. In section 2 we define a rewriting function '*' which transforms non-monotonic properties into conditional theorems or conditional rules. In section 2, using this translation we give a general equivalence theorem between valid formulae in a conditional model and general properties of a class of a non-monotonic relations. Then representation theorems can be trivially obtained using the completeness theorem of conditional logics. In fact one of the main interests of conditional logics is that, given a set of axioms and rules, it is very easy to find a sound and complete semantics in terms of the selection function. It is also possible to transform this latter into another different semantics such as sphere semantics, or relational ones. Section 3 contains an example about how to apply our general representation theorem to preferential systems. We compare the model theory obtained via our approach with those given in the literature.

2 CONNECTIONS

Firstly, in order to illustrate the close connection between conditional logics and non-monotonic inference relations

[4]To obtain the correspondence, the conditional operator is translated into modal language by $(A\Rightarrow B) = \Box(A\rightarrow\Diamond(A\wedge\Box(A\rightarrow B)))$.

we will visually compare some rules used to express properties of non-monotonic deduction relation and some conditional axioms. For example: Reflexivity ($A \vdash A$) and ID-axiom ($A \Rightarrow A$), but also Left Logical Equivalence $\left(\frac{\vdash A \leftrightarrow B \quad A \mid\sim C}{B \mid\sim C}\right)$ and RCEA $\left(\frac{\vdash A \leftrightarrow B}{\vdash A \Rightarrow C \rightarrow B \Rightarrow C}\right)$, Equivalence $\left(\frac{A \mid\sim B \quad B \mid\sim A \quad A \mid\sim C}{B \mid\sim C}\right)$ and CSO $((A \Rightarrow B) \wedge (B \Rightarrow A)) \rightarrow ((A \Rightarrow C) \leftrightarrow (B \Rightarrow C))$, Rational Monotonicity $\left(\frac{A \mid\sim C \quad A \not\mid\sim \neg B}{A \wedge B \mid\sim C}\right)$ and CV $((A \Rightarrow C) \wedge \neg(A \Rightarrow \neg B)) \rightarrow (A \wedge B \Rightarrow C))$.

2.1 A FORMAL CONNECTION

All the previous examples show that there is at least a strong similarity between properties of non-monotonic deduction relation and conditionals. Some authors have already proved that this is more than a similarity, and that some formal correspondence can be found. For example, in [Lamarre 91] a formal equivalence is proved using semantics between preferential non-monotonic inference relations as defined in [Kraus 90] and a particular conditional logic, named C4. Bell [Bell 90] [Bell 91], Katsuno and Satoh [Katsuno 91] have sketched some correspondences including a large class of non-monotonic inference relations. Here, we give a general, formal and complete proof of that correspondence: given a definition of a non-monotonic class of inference relations, it will be possible to obtain the exact corresponding conditional logics and semantics.

Let us begin with some formal definitions of notions commonly used in non-monotonic literature and which will be used in the proof of the equivalence theorem.

`Definition 2.1` A non-monotonic deduction relation '$\mid\sim$' is a set of pairs (A,B) such that A and B are classical formulae. We will note $A \mid\sim B$ when the pair (A, B) belongs to the relation, and $A \not\mid\sim B$ in the opposite case.

A property of non-monotonic inference relations will have the form of a rule $\frac{A_1 \; A_2 \; A_n}{B}$, where each A_i and B are of the form $\vdash X$ or $X_1 \mid\sim X_2$ such that X, X_1 and X_2 are classical formulae. If n=0, the rule is called an axiom. In the sequel we will say that a property (or equivalentely a non-monotonic inference rule) is satisfied by a relation '$\mid\sim$' iff when the premises of the rule belong to the relation or are classical tautologies, then the conclusion belongs to the relation or is a tautology too. It is not difficult to see that all classical operations (contraposition, cut, ...) are available on these rules.

We will now introduce the notion of characteristic set of properties and general properties of a class of non-monotonic inference relations.

`Definition 2.2` A set of properties P is said to be characteristic of a class C of non-monotonic relations iff: all non-monotonic inference relations of C, and only them, satisfy every property in P ($\mid\sim \in$ C iff $\mid\sim$ satisfies every properties in P).

`Definition 2.3` A property p will be said to be a general property of a class C of non-monotonic relations iff: every non-monotonic inference relation of C satisfies p (for every $\mid\sim \in$ C, $\mid\sim$ satisfies p).

In order to prove that the visual similarity is an equivalence, we must define the kind of rules used to describe the properties of non-monotonic deduction relations:

`Definition 2.4` A non-monotonic property p will be said to be of

- type 1 if and only if the monotonic deduction symbol ($\vdash$) does not appear in it;
- type 2 if and only if the monotonic deduction symbol appears only into its hypotheses;
- type 3 if and only if the monotonic deduction symbol appears into its conclusion.

As we will see in the following, properties of type 1 correspond to axioms of conditional logics, properties of type 2 correspond to conditional inference rules, and finally, properties of type 3 do not correspond to anything ever used in logic in general. This last type can be considered as expressing a crucial difference between conditional logics and non-monotonic inference relations.

In order to state formally the connection between non-monotonic deduction relations and conditional logics, we will replace the non-monotonic deduction symbol by the conditional operator. Let us call this operation "*". In the following, two languages and a rewriting operation between them are defined.

`Definition 2.5` Let us define $\mathcal{L}_{NM}$ to be classical propositional language ($\wedge, \vee, \neg, \rightarrow, \ldots$) and metalanguage ($\vdash$) plus the usual non-monotonic symbol $\mid\sim$ connecting classical formulae. $\mathcal{L}_{NM}$ is the language used to express non-monotonic properties. $\mathcal{L}_{Cond}$ is defined adding the conditional connective "$\Rightarrow$" to the classical language.

`Definition 2.6` Let F be an expression of $\mathcal{L}_{NM}$. F* will be the rewritten expression of F in $\mathcal{L}_{Cond}$ obtained in the following way:

- for every classical expression F, $F^* \equiv F$
- for every expression $F \equiv A \mid\sim B$, $F^* \equiv A \Rightarrow B$
- for every expression $F \equiv A \not\mid\sim B$, $F^* \equiv \neg(A \Rightarrow B)$
- for every property of type 1 $F \equiv \frac{A_1 \; A_2 \ldots A_n}{C}$ ($n \geq 0$), $F^* \equiv (A_1^* \wedge A_2^* \wedge \ldots \wedge A_n^*) \rightarrow C^*$
- for every property of type 2,

$F \equiv \dfrac{\vdash B_1 \vdash B_2 \ldots \vdash B_n \; A_1 \; A_2 \ldots A_m}{C}$ (n>0 and m≥0), then $F^* \equiv \dfrac{\vdash B_1 \vdash B_2 \ldots \vdash B_n}{\vdash A_1^* \wedge A_2^* \wedge \ldots \wedge A_m^* \rightarrow C^*}$

- for every property of type 3,

$F \equiv \dfrac{\vdash B_1 \vdash B_2 \ldots \vdash B_n \; A_1 \; A_2 \ldots A_m}{\vdash C}$ (n≥0 and m≥0)

then $F^* \equiv \dfrac{\vdash B_1 \vdash B_2 \ldots \vdash B_n \quad \not\vdash C}{\vdash \neg (A_1^* \wedge A_2^* \wedge \ldots \wedge A_m^*)}$

Note that, as said previously, if p is of type 1 or 2, the form of the obtained expressions (respectively axioms and inference rules) are well-known but if it is of type 3, neither an axiom nor an inference rule is obtained. In fact, it is a negative inference rule which logicians never use. An example of property of this kind is "consistency preservation": $\dfrac{A \not\sim \bot}{A \not\vdash \bot}$ [Gärdenfors 91a]. In the following we focus on properties of type 1 or 2.

2.2 EQUIVALENCE THEOREM

This section contains the main result of this paper. We establish an equivalence theorem between general properties of a class of non-monotonic relations and valid expressions of a corresponding conditional logic obtained by the translation '*'.

Let $\mathcal{P}1,2$ be a set of characteristic properties of type 1 or 2, defining a class of non-monotonic relations C, and containing all classical tautologies. We define $\mathcal{P}^*1,2$ as a logical system containing all the axioms of classical logic, closed under classical inference rules and under the translations by '*' of the non-monotonic properties o f $\mathcal{P}1,2$.

Theorem 2.1 Let S be a semantics which has been proved sound and complete for the system $\mathcal{P}^*1,2$.

Then p is a general property of $\mathcal{P}1,2$ iff its translation p* is valid in S.

Proof:

Part 1: if p is a general property of $\mathcal{P}1,2$, then p* is valid in $\mathcal{P}^*1,2$.

Suppose that p* is not valid in S. Then there exists a model M of S such that M does not satisfy p. Considering this model, we will define a non-monotonic relation $\vdash_M$ such that (1) it belongs to the class of non-monotonic relation C and (2) it does not satisfy p. Let's construct it as follows: A $\vdash_M$B iff M⊨A⇒B.

(1) $\vdash_M$ belongs to the class C If it were not the case, there would exist some characteristic property of C, call it p', such that $\vdash_M$ would not satisfy p'. By definition of $\mathcal{P}1,2$, p' is of type 1 or 2.

(1.a) If p' is of type 1 $\left(\dfrac{A_1 \vdash B_1 \ldots A_n \mid\sim B_n, \ldots C_1 \mid\not\sim D_1 \ldots, C_m \mid\not\sim D_m}{E \vdash F}\right)$. Then, by hypothesis, there are some formulae such that the hypothesis of the rule are true, but not the conclusion. Thus the pairs $(A_1, B_1) \ldots (A_n, B_n)$ belong to the relation $\mid\sim_M$, the pairs $(C_1, D_1), \ldots, (C_m, D_m)$ do not belong to it but the pair (E, F) does not belong to $\vdash_M$. So by definition of $\vdash_M$: $M \models A_1 \Rightarrow B_1$, ..., $M \models A_n \Rightarrow B_n$, $M \models \neg(C_1 \Rightarrow D_1)$,. ..., $M \models \neg(C_m \Rightarrow D_m)$, but $M \not\models E \Rightarrow F$. Furthermore, the axiomatic of the system $\mathcal{P}^*1,2$ is obtained by translating characteristic properties of $\mathcal{P}1,2$. So p'* is an axiom of this system, Then $M \models (A_1 \Rightarrow B_1 \wedge A_n \Rightarrow B_n \wedge \neg(C_1 \Rightarrow D_1) \wedge \neg(C_m \Rightarrow D_m)) \rightarrow (E \Rightarrow F)$, which contradicts the fact that $M \not\models E \Rightarrow F$.

(1.b) p' is a property of type 2 $\dfrac{\vdash G_1 \ldots \vdash G_n \; A_1 \mid\sim B_1 \ldots A_n \mid\sim B_n \; C_1 \mid\not\sim D_1 \ldots C_m \mid\not\sim D_m}{E \vdash F}$.

Then there are some formulae such that the hypotheses of the rule are true but not the conclusion. The only difference to note with the previous proof is that if the G_i are classical tautologies, they are also tautologies in the conditional system which includes classical logic.

(2) $\vdash_M$ does not satisfy the general property p.

(2.a) p is of type 1. By hypothesis p* is not valid in the system $\mathcal{P}^*1,2$. So there is a model M such that $M \not\models p^*$. So the model M satisfies all the premisses of p but M does not satisfy its conclusion. Then by definition of the relation $\vdash_M$, the general property p is not verified by $\vdash_M$.

(2.b) p is of type 2. The proof is similar to the one of the previous step.

So there exists a non-monotonic relation ($\vdash_M$) which

(1) belongs to the class C, defined by the considered set of caracteristic properties for $\mathcal{P}1,2$, and

(2) does not satisfies p. So p is not a general property of $\mathcal{P}1,2$.

Part 2: if p* is valid in $\mathcal{P}^*1,2$, then p is a general property of $\mathcal{P}1,2$.

We will assume that p is not a general property and then prove that p* is not valid.

If p is not a general property then there is a relation $\vdash$, belonging to the class C defined by $\mathcal{P}1,2$, such that $\mid\sim$ does not satisfy p. Let F be a set of formulae defined by: F= $\{A \Rightarrow B : A \mid\sim B\} \cup \{\neg(A \Rightarrow B) : A \mid\not\sim B\}$.

(1) The set F is closed under the rules and axioms of the system $\mathcal{P}^*1,2$ as far as flat conditionals are concerned. If it were not the case, there would be a formula $A \Rightarrow B$ or $\neg(A \Rightarrow B)$ which could be deduced, using the system $\mathcal{P}^*1,2$. The axiom or the inference rule used to draw this conclusion is a translation of a general property of $\mathcal{P}1,2$. And so, this rule is not satisfied by the relation $\vdash\!\sim$. This is a contradiction with the fact that $\vdash\!\sim$ belongs to $\mathcal{P}1,2$.

(2) F is a consistent set of formulae. If it were not the case, there would be a pair of formulae $A \Rightarrow B$, and $\neg(A \Rightarrow B)$ which could be deduced form F. By (1) they would belong to F, and then by definition of F: $A \vdash\!\sim B$ and $A \not\vdash\!\sim B$, which is impossible by definition of a non-monotonic inference relation.

So there is a model M satisfying F, and such that it does not satisfy p*. So p* is not valid.

The previous theorem enables us to write representation theorems using semantics of conditional logics for many deduction relation classes.

From a logical point of view, note that theorem 2.1 gives us some results on "validity" (truth in every interpretation). So representation theorems obviously follow from it since they use a weaker notion to represent a non-monotonic deduction relation: "satisfiability".

Corollary 2.1: General Representation Theorem.

Let $\mathcal{P}1,2$ be a consistent set of properties defined as previously. Let $\mathcal{P}^*1,2$ be the associated conditional logic, and assume that some completeness theorem has been established.
Then $\vdash\!\sim$ is a deduction relation verifying all properties of $\mathcal{P}1,2$ if and only if there is an interpretaion M of $\mathcal{P}^*1,2$ such that $A \vdash\!\sim B$ iff $M \models A \Rightarrow B$.

Note that the most general semantics known for conditional logics is the semantics based on selection function as defined in [Lewis 86] and generalised in [Nute 80]. In this semantics a model is a triple <W, f, v> where W is a non-empty set of possible worlds, v a valuation function, and f a selection function which associates to each couple (world, subset_of_possible_words) (a formula is caracterised by the set of worlds satisfying it in the model) a set of possible worlds. This semantics can be used for a very large class of conditional logics: all normal conditional logics. So it would be a powerful tool to represent and study non-monotonic inference relations.

A comparison of the result presented here with other works on the same subject will lead us to some questions. Why to use a semantics based on a selection function, and not a semantics based on an accessibility relation over possible worlds as in [Katsuno 91]? The answer to this question is that we prefer the semantics which is able to capture as many systems as possible. The semantics based on a selection function is probably more difficult to manipulate, but it is also more general. Indeed, in our opinion it would be very difficult to find a semantics using an accessibility relation for conditional logics strictly included in the one corresponding to cumulative systems. Another advantage of a semantics using a selection function is that a condition corresponds to each particular axiom independently of the context, in such a way that given a set of axioms, it is very easy to find the corresponding semantics.

Furthermore, the index is necessary if we are interested in the exact semantics corresponding to a conditional logic associated to a non-monotonic inference system. Even if a restricted language is considered (Boolean combinations of flat conditionals) the semantics continues to deal with formulae which have been removed from the language. This introduces a gap between syntax and semantics. This gap explains why so many different semantics can be used to give representation theorems for the same non-monotonic systems. The following example shows how it is possible to find a conditional logic with a full correspondence. Some examples illustrating this gap will be discussed at the end of the following section.

3 APPLICATION TO PREFERENTIAL SYSTEMS

The following example shows how to apply the connection presented in chapter 2, to find a simple representation theorem for the preferential inference system.

`Definition 3.1` [Kraus 90] A consequence relation is a *preferential relation* if and only if it satisfies all the instances of:

Reflexivity axiom $\quad A \vdash\!\sim A$

and is closed under the inference rules of

Left Logical Equivalence: $\dfrac{\vdash A \leftrightarrow B \quad A \vdash\!\sim C}{B \vdash\!\sim C}$

Right Weakening: $\dfrac{A \vdash\!\sim B \quad \vdash B \rightarrow C}{A \vdash\!\sim C}$

Cut: $\dfrac{A \wedge B \vdash\!\sim C \quad A \vdash\!\sim B}{A \vdash\!\sim C}$

Cautious Monotonicity: $\dfrac{A \vdash\!\sim B \quad A \vdash\!\sim C}{A \wedge B \vdash\!\sim C}$

Or: $\dfrac{A \vdash\!\sim C \quad B \vdash\!\sim C}{A \vee B \vdash\!\sim C}$

Applying the rewriting operation '*', we obtain :

$\vdash A \Rightarrow A$

$\vdash ((A \Rightarrow B) \wedge (A \wedge B \Rightarrow C)) \rightarrow (A \Rightarrow C)$

$\vdash ((A \Rightarrow B) \wedge (A \Rightarrow C)) \rightarrow (A \wedge B \Rightarrow C)$

$\vdash ((A \Rightarrow C) \wedge (B \Rightarrow C)) \rightarrow (A \vee B \Rightarrow C)$

$$\frac{\vdash A \leftrightarrow B}{\vdash A \Rightarrow C \leftrightarrow B \Rightarrow C}$$

$$\frac{\vdash(B\rightarrow C)}{\vdash(A\Rightarrow B)\rightarrow(A\Rightarrow C)}$$

Adding this set of rules and axioms to classical logic gives us an equivalent axiomatisation for the conditional logic WC§, (the logic WC defined in [Nute 80] without the conditional axiom MP $(A\Rightarrow B)\rightarrow(A\rightarrow B)$) [Crocco 92b]. The semantics associated with this system is given by:

Definition 3.2 A model of WC§ is a triple $M=<W,f,m>$ where : W is a non-empty set of worlds, m is a meaning function and f is a selection function $(W\times 2^W\rightarrow 2^W)$ satisfying the conditions :

- $f(w,|A|)\subseteq|A|$ [5]
- if $f(w,|A|)\subseteq|B|$ then $f(w,|A|)=f(w,|A\wedge B|)$.
- $f(w,|A\vee B|)\subseteq f(w,|A|)\cup f(w,|B|)$

The satisfiability relation ($\models$) is defined as usual, in particular $M,w\models A\Rightarrow B$ iff $f(w,|A|)\subseteq|B|$

Representation theorem 3.1
A deduction relation $\vdash$ is a preferential relation if and only if there is some WC§ model M and some world w such that $A\vdash B$ iff $M,w\models A\Rightarrow B$.

The proof is direct from corollary 2.1 and the completeness theorem for WC§.

Katsuno and Satoh [Katsuno 91] have proposed another axiomatic to obtain a correspondence between preferential deductions and conditional logics. Their system (TO) includes the axiom MP: $(A\Rightarrow B)\rightarrow(A\rightarrow B)$ which is not necessary. Indeed, MP is not a theorem in the system WC§ which is proved to be sufficient to give a representation theorem (Theorem 3.1). Furthermore, using the contrapositive, it is possible to prove $\neg(A\rightarrow B)\rightarrow\neg(A\Rightarrow B)$, and then using some substitution: $\top\rightarrow\neg(\top\Rightarrow\bot)$, or more simply $\neg(\top\Rightarrow\bot)$. But the corresponding non-monotonic property $\top\not\vdash\bot$ is not a general property of preferential relation. The only preferential relation satisfying this property is the trivial one ($A\vdash B$ for any formulae A and B). Since this relation does not present a real interest, it is not a very important problem, and the semantics based on a partial preorder over interpretations is easier to manipulate.

Bell [Bell 91] argues that Preferential Inference Relations correspond to the system S1 of Burgess [Burgess 81] but also to the system P of Veltman [Veltman 85] and to his system C [Bell 90]. This last correspondence is doubtful as the system C contains the axioms MP and CS: $(A\wedge B)\rightarrow(A\Rightarrow B)$. This last axiom, even if the language is restricted only to Boolean combinations of flat conditionals, has some bad consequences. Indeed, using the classical tautology $(\top\wedge A)\vee(\top\wedge\neg A)$, and CS it is possible to derive $(\top\Rightarrow A)\vee(\top\Rightarrow\neg A)$. But there are some non-monotonic inference relations which do not satisfy the non-monotonic property $\frac{\top\not\vdash\neg A}{\top\vdash A}$ corresponding to this C-theorem.

In a similar way, if we consider the correspondence established by Boutilier [Boutilier 90] or in a [Lamarre 91], the axiom $(\neg A\Rightarrow A)\rightarrow A$ is valid in both of these systems. This is probably very intuitive considering the semantics associated to these kinds of inference relations, via a representation theorem given by Kraus and al. [Kraus 90] which is very near to the semantics of S4. But addding this axiom is not necessary. As MP, this axiom allows to deduce $\top\not\vdash\bot$ which, as we already said, is not a general property of preferential inference relations. Furthermore, the conditional logic presented in [Lamarre 91] uses nested conditional axioms. They are necessary to obtain soundness and completeness with S4-semantics, but, none of them are valid in WC§ which is the minimal conditional logic to obtain such a correspondence.

All these comments illustrate the remark at the end of paragraph 2. There are many different possibilities (at most five proposed by different authors) to establish representation theorems. Only one is minimal, with a full correspondence on a full language.

4 CONCLUSION

Conditionals can be seen, then, as a unifying frame for non-monotonic deduction relation properties. They allow to give very simple representation theorems, a clear syntactical monotonic counterpart of the non-monotonic relations classes[6], a simple classification for all the systems defined in the literature. They can be used to define new non-monotonic relations class especially for those included in the cumulative ones. In fact there is a large set of conditional logics that can be defined, strictly included in the conditional logic associated to cumumative inference relations.

Nevertheless there are at least two criticism to the semantics proposed in this paper. Firstly, it seems that the conditional models for the preferential inference relations are not as simple as the semantics based on preferential relations over states. This is probably true but connections have been proved between some conditional logics and normal modal logics (semantics of which are based on relations between worlds). So a systematic method could be found to obtain, given a conditional logics, the corresponding semantics in terms of properties of an accessibility relation over worlds. This is not surprising as there is an intuitive relation between the necessary operator and the conditional one (see [Chellas 75]). Furthermore, theorem 2.1 allows to use any semantics provided that it have been proved sound and

[5] Where $|X|$ is the subset of possible worlds satisfying X.

[6] Deduction systems for some Conditional logics have been defined in a tableaux like presentation [Lamarre 92] and in a sequent system form [Crocco 92a].

complete with the considered axiomatic. There is no restriction on the form of this semantics.

Secondly, there is no direct syntactical conditional translation of properties of type 3, such as consistency preservation. However, it is possible to add a condition to these models to satisfy these properties.

Our opinion is, for the two objections, that the advantage of having a unifying semantical tool for representation theorems, giving a full equivalence between models and non-monotonic relations, is not negligible.

Acknowledgements

We would like to thanks Luis Fariñas del Cerro and Costa Dosen for many valuable discussions on the subject. We are also greatful to Sylvie Cazalens and Jerome Lang for their comments.

References

[Bell 90] Bell, J., "The Logic of Nonmonotonicity", *Artificial Intelligence*, vol. 41, pp. 365-374, 1990.

[Bell 91] Bell, J., "Pragmatic Logics", in *KR'91* Allen, J., Fikes, R., and Sandewall, E., Ed., Morgan Kaufmann, April 1991, pp. 50-60.

[Boutilier 90] Boutilier, G., "Conditional Logics of Normality as Modal Systems", in *AAAI*, MIT Press, 1990, pp. 594-599.

[Burgess 81] Burgess, J. P., "Quick completeness proofs for some logics of conditionals", *Notre Dame Journal Formal Logic*, vol. 22, no. 1, pp. 76-84, 1981.

[Chellas 75] Chellas, B.F., "Basic Conditional Logics", *Journal of Philosophical Logic*, no. 4, pp. 133-153, 1975.

[Crocco 92a] Crocco, G. Fariñas del Cerro L., "Some Idears for the Definition of Structural Logics", Technical Report IRIT/92-2-T, IRIT - Université Paul Sabatier -118 route de Narbonne - 31062 Toulouse - France, 1992.

[Crocco 92b] Crocco, G. and Lamarre, P., "Conditional Logics: a Powerful Tool for Non Monotonicity", Technical Report IRIT 92-2-R, IRIT - Université Paul Sabatier -118 route de Narbonne - 31062 Toulouse - France, Janvier 1992.

[Delgrande 88] Delgrande, J.P., "An approach to Default Reasoning Based on First Order Conditional Logic: Revised Report", *Artificial Intelligence*, no. 36, pp. 63-90, 1988.

[Freund 91] Freund, M., "A Semantical Characterisation of Disjunctive Relations", in *Lecture Notes in Artificial Intelligence*, Springer Verlag, Ph. Jorrand and J. Kelemen, 1991, pp. 72-83.

[Gabbay 85] Gabbay, D.M., "Theoritical Foundations for Non-Monotonic Reasoning in Expert Systems", in *Logics and Models of Concurent Systems*, Apt, K.R., Ed.. Springer Verlag, 1985, pp. 439-457.

[Gärdenfors 91a] Gärdenfors, P., "Nonmonotonic inference based on expectation: A preliminary report", in *KR'91.*, Allen, J., Fikes, R., and Sandewall, E., Ed., Morgan Kaufmann, April 22-25 1991, pp. 585-590.

[Gärdenfors 91b] Gärdenfors, P. and Makinson, D., "Nonmonotonic Inference Based on Expectations", .

[Katsuno 91] Katsuno, H. and Satoh, K., "A Unified View of Consequence Relation, Belief Revision and Consequence Logic", in *Proceedings of the twelfth International Join Conference on Artificial Intelligence (I.J.C.A.I.).*, vol. 1, Morgan Kaufmann, August 1991, pp. 406-412.

[Kraus 90] Kraus, S., Lehmann, D., and Magidor, M., "Nonmonotonic Reasoning, Preferential Models and Cumulative Logics", *Artificial Intelligence* , no. 44, pp. 167-207, 1990.

[Lamarre 91] Lamarre, P., "S4 as the Conditional Logic of Nonmonotonicity", in *KR'91*, Allen, J.A., Fikes, R., and Sandewall, E., Ed., Morgan Kaufmann, San Mateo CA, 1991, pp. 357-367.

[Lamarre 92] Lamarre, P., "A Promenade from Monotonicity to Non Montonicity Following a theorem Prover", *KR'92*, 1992.

[LeaSombe 89] LéaSombé, *Raisonnement sur des Informations Incomplètes en Intelligence Artificielle.* Tekna, 1989.

[Lehmann 89] Lehmann, D., "What does a Conditional Knowledge Base Entail", in *KR'89*, R.J. Brachman, H.J. Levesque, R.R., Ed., Morgan Kaufmann, 1989, pp. 212-222.

[Lewis 86] Lewis, D., *Counterfactuals.* Basil Blackwell Ltd, 1986 (first published 1973).

[Nejdl 91] Nejdl, W., "The P-Systems: A Systematic Classification of Logics of Non Monotonicity", in *Proceedings Night National Conference on Artificial Intelligence (AAAI)*, vol. 1, AAAI Press / The MIT Press, July 14-19 1991, pp. 366-372.

[Nute 80] Nute, D., *Topics in Conditional Logics.* D. Reidel Publishing Company, 1980.

[Shoham 88] Shoham, Y., *Reasoning about Change*. The MIT Press, 1988.

[Veltman 85] Veltman, F.J.M.M., "Logics for Conditionals", Ph.D. thesis, University of Amsterdam, 1985.

A Promenade from Monotonicity to Non-Monotonicity Following a Theorem Prover

Philippe Lamarre
IRIT - Université Paul Sabatier
118, route de Narbonne
31062 Toulouse Cedex - France
Email: lamarre@irit.fr

Abstract

Normal conditional logics have been shown to be appropriate to formalize a large number of common-sense assertions such as prototypal properties, obligation, possibility, non-monotonic inference relations, ... The basic language of these logics is the classical one augmented with a conditional operator '⇒' which has been first introduced to capture sentences of the form "If it were the case that ..., then it would be the case that ...". In this paper, we present a tableau-like theorem prover for a subclass of these logics. Given a formula, the approach is to find an assignment falsifying it. It is shown sound and complete with the considered systems. The particular model computed via the procedure (coutermodel) is then used to define a non-monotonic inference relation which is proved to be equivalent to Pearl's system Z. Then a mutation changes our theorem prover into a non-monotonic prover.

1 INTRODUCTION.

It is now generally admitted that a large variety of common sense expressions cannot be captured within classical logic. For example, defaults rules, prototypical properties, revision and possibility notions cannot be represented very easily using material conditional. Conditional logics have been proved to be a powerful tool for these notions ([Chellas 75] [Lewis 86], [Nute 80,84], [Delgrande 88,89], [Nejdl 91], [Fariñas 91]...). The main principle of these logics is to introduce a new operator, '⇒' called variable conditional, in addition to classical connectives. It is exactly the same method which is used in modal logics when the classical language is augmented with a necessity operator. As in modal logics, the semantics is usually given in terms of possible worlds. The truth of the conditional is linked not only to the actual world, but also to a set of possible worlds which are considered as interesting according to the conditional hypothesis, and the actual world.

This paper presents a method determining the validity of formulae in a subclass of conditional logics. The main idea is to attempt to build a model for the negation of the considered formula, like in the semantic tableau method [Smullyan 68] [Hughes 68]. If such a model can be obtained, then the formula is not valid. In the other case, it is valid, or in other terms, using soundness and completeness, it is a theorem.

In next section, we will briefly recall what are normal conditional logics (axiomatic, semantics, meaning of the conditional operator). In section 3, the theorem prover is presented, in two different forms: in an algorithmic form, and in a tableau-like presentation. Using some well-known correspondences between conditional logics and non-monotonic inference relations, section 4 shows how this theorem prover can be used to study common properties of non-monotonic rational inference relations. Finally, in section 5, we will mute our prover into a non-monotonic one.

2 NORMAL CONDITIONAL LOGICS

The family of formal systems named "Conditional Logics" has been introduced at the end of the 60th in the aim of giving a formal account of linguistic structures of the form "If it were the case that ..., then it would be the case that ...". Different analyses of these sentences, named "Couterfactuals", give different systems [Lewis 86] [Stalnaker 68] [Nute 80]. Generally, these sentences are close to revision notions. Indeed, as elaborated by Stalnaker [Stalnaker 68], Ramsey's rule for evaluating conditionals is:

> *... first, add the antecedent (hypothetically) to your stock of beliefs; second, make whatever adjustments are required to maintain consistency (without modifying the hypothetical belief in the antecedent); finally, consider whether or not the consequent is then true.*

But couterfactuals can be understood in some other ways. Indeed, the first part of a couterfactual "If it were the case that ...", may refer, not to the situation(s) obtained by revising the actual one, but to the most normal one(s) according to the antecedent. With this understanding, the couterfactual may also be read "Normally, when it is the case that ..., then it is also the case that ...". This way has been explored by Delgrande [Delgrande 87].

The gap between these two approaches is not large, and there are a lot of bridges. The language is common (propositional one augmented with a binary conditional operator $\Rightarrow$). All conditional logics used to formalize these approaches are normal, i.e. they contain the system CK.

Definition 1 *System CK* [Chellas 75]
The minimal normal conditional logic CK is the smallest system which contains the classical propositional axioms, and which is closed under Modus Ponens, and the following two rules:

$$\text{RCEA: } \frac{\vdash \alpha \leftrightarrow \beta}{\vdash (\alpha \Rightarrow \chi) \leftrightarrow (\beta \Rightarrow \chi)}$$

$$\text{RCK: } \frac{\vdash \beta_1 \wedge \ldots \wedge \beta_n \rightarrow \beta}{\vdash ((\alpha \Rightarrow \beta_1) \wedge \ldots \wedge (\alpha \Rightarrow \beta_n)) \rightarrow (\alpha \Rightarrow \beta)} (n \geq 0).$$

And, another important common point is that the semantics of these logics can be given in terms of possible worlds and selection function.

Definition 2 *CK-Model* [Chellas 75]
A CK-Model is a triple $M = \langle \mathcal{W}, f, v \rangle$, such that $\mathcal{W}$ is a non-empty set of worlds, v is a valuation function as in Kripke models for modal logics, and f is a selection function: $\mathcal{W} \times 2^{\mathcal{W}} \rightarrow 2^{\mathcal{W}}$.

The satisfiability relation is defined as usual for classical operators, and as expected for the conditional operator: $M,w \models \alpha \Rightarrow \beta$ iff $f(w, |\alpha|^M) \subseteq |\beta|^M$ where $|\chi|^M$ (or simply $|\chi|$ when there is no ambiguity on M) is the set of worlds of $\mathcal{W}$ satisfying χ (or χ-worlds). Intuitively, the selection function picks up the worlds which are the most interesting ones according to the actual world and the considered hypothesis. The main interest of this semantics is that, for each conditional axiom, there exists a semantical condition on f which exactly corresponds to it. Moreover, it can be applied to a large class of conditional logics.

In the following, we will focus on a particular class on conditional logics: logics presented by Lewis [Lewis 86] to capture revision and the logic proposed by Delgrande to deal with normality [Delgrande 87]. These logics have the system V as a common part:

Definition 3 *System V* [Lewis 86]
The system V is the smallest system containing CK and the following axioms:
ID: $\alpha \Rightarrow \alpha$
CSO: $((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \rightarrow ((\alpha \Rightarrow \chi) \leftrightarrow (\beta \Rightarrow \chi))$
CA: $((\alpha \Rightarrow \chi) \wedge (\beta \Rightarrow \chi)) \rightarrow ((\alpha \vee \beta) \Rightarrow \chi)$
CV: $((\alpha \Rightarrow \chi) \wedge \neg(\alpha \Rightarrow \neg\beta)) \rightarrow ((\alpha \wedge \beta) \Rightarrow \chi)$

As we said, the semantics of this system can be given in terms of a selection function. To each of these axioms corresponds a semantical condition on f: id: "$f(w, |\alpha|) \subseteq |\alpha|$"; cso: "If $f(w, |\alpha|) \subseteq |\beta|$ and $f(w, |\beta|) \subseteq |\alpha|$ then $f(w, |\alpha|) = f(w, |\beta|)$; ca: "$f(w, |\alpha \vee \beta|) \subseteq f(w, |\alpha|) \cup f(w, |\beta|)$"; cv: "If $f(w, |\alpha|) \not\subseteq |\neg\beta|$ then $f(w, |\alpha \wedge \beta|) \subseteq f(w, |\alpha|)$". But the semantics can also be given in term of spheres:

Definition 4 *Sphere semantics* [Lewis 86]
A sphere model is a triple $M = \langle \mathcal{W}, S, v \rangle$, such that $\mathcal{W}$ and v are respectively a non-empty set of worlds and a valuation function; and S is a function $\mathcal{W} \rightarrow 2^{2^{\mathcal{W}}}$ which associates to each world w of $\mathcal{W}$ a sphere system S(w) such that:
P1 - *Nested*: If $s_1 \in S(w)$ and $s_2 \in S(w)$ then $s_1 \subseteq s_2$ or $s_2 \subseteq s_1$,
P2 - S(w) is *closed under union*,
P3 - S(w) is *closed under non-empty intersection*.

Note that there is no link between a world and its associated sphere system. Furthermore, there is no constraint between two sphere systems so that nested formulae do not have any particular property.

The main interest of this semantics is that models can be drawn on a graphic in such a way that they are much easier to manipulate.

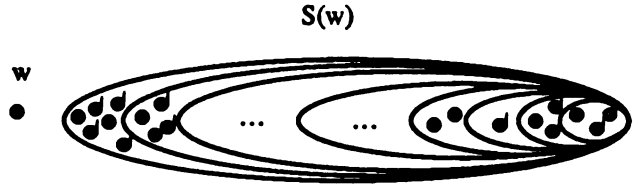


Figure 1: Sphere system

From a revision point of view, considering w as the actual world, the intuitive meaning is that the closer to w is some world, the smaller is the sphere to which it belongs. In a similar fashion, from a normality point of view, the more normal is a world, the smaller is the sphere to which it belongs. According to these intuitive readings of a sphere system, the satisfiability relation is defined by:

Definition 5 *Satisfiability relation in a sphere model* [Lewis 86]
In a sphere model, the satisfiability relation is defined as usual for the classical operators. The only interesting case is for the conditional operator: If M is a sphere model, and w one of its worlds: $M,w \models \alpha \Rightarrow \beta$ iff (1) $\alpha \rightarrow \beta$ is true in all the worlds of the smallest sphere which contains at least one world w' satisfying α, or (2) there is no world satisfying α in any sphere.

This definition implies that each time there is a world satisfying α in some sphere, there is a smallest sphere containing at least one α-world. Indeed, system V is sound and complete with respect to such models [Lewis 86], but this means that models with infinite sequences are not allowed. To be able to consider such models, just replace the condition (1) of the conditional's satisfiability definition by "there is some sphere containing an α-world such that all its worlds satisfy $\alpha \rightarrow \beta$". These two definitions are clearly equivalent considering finite models, but the second is more general since it deals with infinite sequences.

Now, just consider how revision and normality approaches differ. On the one hand, characteristic axioms generally added to obtain some revision notions are:

N: $\neg(\top \Rightarrow \bot)$;

MP: $(\alpha \Rightarrow \beta) \rightarrow (\alpha \rightarrow \beta)$;

CS: $(\alpha \wedge \beta) \rightarrow (\alpha \Rightarrow \beta)$.

Their associated semantical conditions are respectively:

n: sphere systems are not empty, $\bigcup_{s \in S(w)} s \neq \emptyset$,

"weakly centered": $S(w) \neq \emptyset$ and If $s \in S(w)$ then $w \in s$,

"likely centered": If $\bigcup_{s \in S(w)} s \neq \emptyset$ then $\{w\} \in S(w)$.

When both MP and CS are present, the system is said "Centered": $\{w\} \in S(w)$.

Lewis defines systems VN, VW and VC by adding to V respectively N, MP, and (MP and CS).

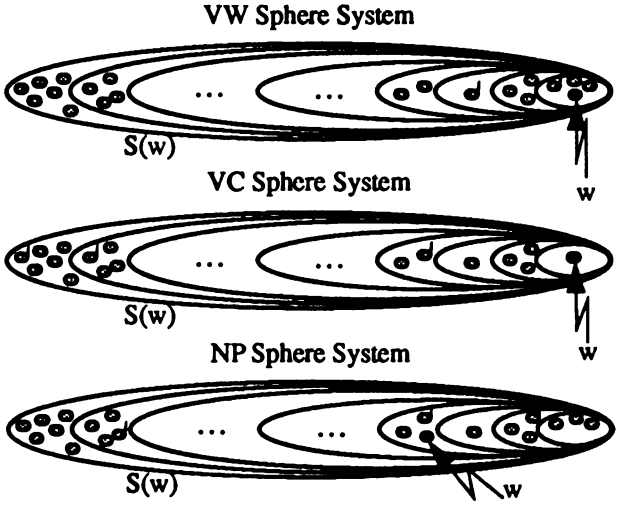


Figure 2: Sphere systems.

On the other hand, Delgrande's conditional logics NP [Delgrande 87], dealing with normality notions, may de defined as V plus the axiom $(\neg\alpha \Rightarrow \alpha) \rightarrow \alpha$[1] the semantical condition of which, in sphere terms, is: there is some sphere s such that $s \in S(w)$, and $w \in s$.

CK - - - → V → VN → NP → VW → VC

Figure 3: Systems hierachy

[1]This axiom does not appear explicitly in [Delgrande 87], but the author argue that semantics of NP can be given in terms of possible worlds and accessibility relation, and that the corresponding modal logic is S4.3. Since in this context $\neg\alpha \Rightarrow \alpha$ is nothing else than $\Box\alpha$ (necessary α), this axiom must be present since it correspond to modal axiom T of S4.3. Note that, in a next paper [Delgrande 89], the author has added this axiom to this logic.

3 A THEOREM PROVER FOR A CLASS OF CONDITIONAL LOGICS

In this part, we will first present a theorem prover for the conditional logic V which is the common basis of the two approaches presented above. In a second step, we will see that it is not difficult to modify this prover in order to obtain theorem provers for VN, NP, VW, VC, ...

Our theorem prover is founded on the semantics, and as usual for such methods, to prove that a formula α is a theorem, we prove that no model exists for its negation. So, given a formula $(\neg\alpha)$, the problem is to find one of its models. To simplify a little this problem, let us write this formula in its equivalent "normal disjunctive form": $\neg\alpha = \bigvee_i \bigwedge_j \beta_{i,j}$, where each $\beta_{i,j}$ is either a literal or a conditional formula (i.e.: a formula with the conditional operator as main operator $(\phi \Rightarrow \psi)$, or a negation of such a formula $\neg(\phi \Rightarrow \psi)$). Since it is not difficult to prove that $\neg\alpha$ is consistent if and only if some of its disjunct is, in the following, we will only consider formulae of the form $\bigwedge_j \beta_j$. Given such a formula, we try to build a model. Clearly, if there is a classical inconsistency, we can conclude that there is no model. But, if there is no classical inconsistency, we have to build a sphere system.

3.1 SYSTEM V: PRELIMINARY NOTIONS.

To build a sphere system, the general principle is to compute it sphere by sphere beginning by the smallest one. A sphere will be characterized by a set of formulae: those which are true in all its worlds. Let us begin by the smallest one[2]. It is computed considering the conditionals of $\bigwedge_j \beta_j$.

- Let β_j be a positive conditional $(\phi \Rightarrow \psi)$. Looking at the satisfiability of such a formula, two cases must be considered. First, if there is a world in some sphere satisfying the hypothesis ϕ, then $\phi \rightarrow \psi$ is true in all the worlds of the smallest sphere which contains at least a world w' satisfying ϕ. Since spheres are nested (P1 of Definition 4), the smallest sphere is included in the smallest sphere containing at least a world w' satisfying ϕ. So $\phi \rightarrow \psi$ is true in all the worlds of the smallest sphere. Second case: there is no world satisfying ϕ in any sphere. So trivially, each world of the smallest sphere satisfies $\phi \rightarrow \psi$.
  To conclude: if $\phi \Rightarrow \psi$ is some β_j then $\phi \rightarrow \psi$ is true in all the worlds of the smallest sphere.
- Let β_j be a negative conditional $\neg(\phi \Rightarrow \psi)$. Considering the definition of the conditional, such a formula is true iff in the smallest sphere containing a ϕ-world (such a sphere must exist), there is a world satisfying both ϕ and $\neg\psi$. Now suppose that in the smallest sphere, there is no world satisfying $\phi \wedge \neg\psi$, then there is no world satisfying ϕ. If it were the case, it would contradict what we have just said about the smallest sphere containing a ϕ-world.

These remarks give us a method to compute the characteristic formulae of the smallest sphere: the Core.

[2]Remember that system V is sound and complete with respect to models without infinite descending chain of spheres.

Definition 6 *Core of a set of formulae.*

Let $\bigwedge_j \beta_j$ be a conjunction of classical literals and conditionals (positive or negative ones).

The set $Core(\bigwedge_j \beta_j)$, is defined by:

$Core_0(\bigwedge_j \beta_j) = \{ \phi \rightarrow \psi : \phi \Rightarrow \psi$ is some β_j in $\bigwedge_j \beta_j \}$

$Core_i(\bigwedge_j \beta_j) = Core_{i-1}(\bigwedge_j \beta_j) \cup \{\neg\phi\}$ if there is some $\neg(\phi \Rightarrow \psi)$ of $\bigwedge_j \beta_j$ such that $\phi \wedge \neg\psi$ is not consistent with $Core_{i-1}(\bigwedge_j \beta_j)$, else $Core_i(\bigwedge_j \beta_j) = Core_{i-1}(\bigwedge_j \beta_j)$.

Then $Core(\bigwedge_j \beta_j) = \bigcup_{i=0}^{\infty} Core_i(\bigwedge_j \beta_j)$

The consistency test which is necessary here will be done using the procedure presented in the next section. Note that at each recursive call, the degree of formulae decreses by one. Then, this difinition is constructive. Indeed, given a set of conditionals, the first part of the core is easily computed considering positive conditionals. The second part corresponds to negative conditionals, and is a little more difficult to compute. If there is some conditional $\neg(\phi \Rightarrow \psi)$ such that $\phi \wedge \neg\psi$ is not consistent with the core as computed at this step, $\neg\phi$ must be added to the core. Then consistency of the hypothesis of the other negative conditionals with this set must be tested. If a formula were not consistent with the previous set, it would not be consistent with this one, but a formula consistent with the previous set may no longer be consistent with the new one. If considered formulae are classical ones, checking consistency does not present any problem. But if a conditional operator occurs (this is possible when nested conditionals appears in the original formula), the method presented here must be applied recursively.

Lemma 1 *Core: a constant set in any model.*

Let $\bigwedge_j \beta_j$ be a conjunction of classical literals and conditionals (positive or negative ones).

Let Core be the set of formulae defined as in Definition 6 considering $\bigwedge_j \beta_j$.

For each model M and each world w such that $M,w \models \bigwedge_j \beta_j$, if s is the smallest sphere of S(w), then each world w' of s satisfies each formula of the set Core.

The proof is by induction on i ($Core_i$ of the definition) [Lamarre 92].

Definition 7 *Captured conditional*

A conditional, $(\phi \Rightarrow \psi)$ or $\neg(\phi \Rightarrow \psi)$ is said captured in a sphere s if and only if there is some world w in s such that $M,w \models \phi$.

Indeed, by definition of the conditional, the truth value of this conditional only depends on this sphere (or included ones), and is not modified whatever the bigger spheres contain. So intuitively, captured conditionals can be forgotten to compute the including spheres

So, considering conditionals, to build the whole model we build it sphere by sphere beginning by the smallest one. Each sphere contains as many worlds as possible such that, to know if a conditional is captured or not, we only have to check if its hypothesis is consistent with the core. And at each step captured conditionals will be picked up.

3.2 SYSTEM V: DECISION PROCEDURE.

```
PROCEDURE V-Consistent;
IN: Γ = Set of conditionals and classical formulae;
OUT: the answer to the question "is Γ consistent";
BEGIN
  D := Conditionals of Γ;
  C := Classical formulae of Γ;
  IF C is consistent[3] THEN
     IF D = ∅ THEN the answer is 'YES'
     ELSE   BEGIN
        REPEAT
          Compute the core N of D;   % as explained in the
                                       previous section%
          Compute the set CD of captured conditionals in
             the sphere characterised by N;
          D := D\CD
        UNTIL (D=∅) OR (CD=∅);
        IF there is some uncaptured negative conditional
          THEN the answer is 'NO'
          ELSE the answer is 'YES'
             END
  ELSE the answer is "NO"
END.
```

Definition 8 *Deduction via the procedure*

Let α be a formula.

$\vdash_{Proc.V} \alpha$ iff for each disjunct of the normal disjunctive form of $\neg\alpha$, the answer of the procedure V-Consistent is 'NO'.

Theorem 1 *Soundness/Completeness*

Let α be a finite[4] formula. Then $\vdash_{Proc.V} \alpha$ iff $\vdash_V \alpha$.

The "if" part, using the contrapositive, is equivalent to: "If $\neg\alpha$ is consistent, then there is a disjunct such that the answer is 'YES'". The method used to prove that point is to consider the disjunctive normal form of $\neg\alpha$ and, considering a disjunct, to prove that "if it is consistent, then the answer is 'YES'". Or, using the contrapositive once more "If the answer is 'NO', then the disjunct is not consistent". This is done inductively on the nested degree of formulae. Trivially, this is true for classical formulae. Now, assuming that this property is true for a degree n, suppose that the answer is 'NO', and that the disjunct (of degree n+1) is satisfiable, i.e. a model M, w of it exists. Considering the procedure, if the answer is 'NO', there is a set of uncaptured conditionals, $\mathcal{UC}$, containing a negative one. By hypothesis: M,w satisfies all of them. Now considering the procedure, no world satisfying $Core(\mathcal{UC})$ can satisfy any conditional hypothesis. So, by Lemma 1, the smallest sphere of the sphere system S(w) does not capture any conditional. Then the model obtained from M removing this smallest sphere is also a model of these conditionals. This can be applied recursively until $S(w) = \emptyset$. But here is a contradiction. Indeed, the final model is a model of $\mathcal{UC}$, but there is some negative conditional in this set, and such a formula cannot

[3] Classical method like tableau method for propositional logic can be used here.

[4] Without this hypothesis, the procedure may not finish in a finite time.

be satisfied (by definition) in an empty sphere system. So the property is also true for n+1 and then for all formulae.

The "only if" part is obtained building a model of a disjunct when the answer of the procedure is 'YES' for a particular disjunct of the normal disjunctive form of $\neg\alpha$. Intuitively this model is as big as possible. We have prove that each formula of the core must be satisfied in all the worlds of the smallest sphere of any model. If we define the smallest sphere as the set of all worlds satisfying the core, it is the maximal smallest sphere. So as many conditionals as possible will be captured in the smallest sphere, and as few conditionals as possible will be considered to compute the core of the including sphere. Then this core will be as small as possible, and the sphere as big as possible, ... By definition, it is not difficult to see that each captured conditional is satisfied in this model, and since there is no uncaptured negative conditional, all conditionals are satisfied. So we have build a model corresponding exactly to the "trace" of the procedure and which satisfies the disjunct.

3.3 ADAPTATION TO OTHER SYSTEMS.

To build theorem provers for the other systems presented here is not difficult. Since their semantics also use sphere systems, the main principle is not changed. We only have to consider the characteristic conditions of these systems to modify the procedure.

Let begin with system VN. Remember that the semantical condition added to obtain the semantics is: $\bigcup_{s\in S(w)} s \neq \emptyset$. So it is not possible to have an empty sphere system. Considering the principle used previously, to obtain an empty sphere system is possible iff the first computed core is not consistent. So to obtain a theorem prover for VN, a consistency test has to be added to the first computed core, and if this set is not consistent, the answer is 'NO'.

For system NP, we have to check the consistency of classical formulae with the core of the biggest sphere. Indeed, if classical formulae are not consistent with the core of the biggest sphere, they are not consistent with the core of any sphere. So, in the procedure, we check the consistency of classical formulae with the core of the uncaptured conditionals. If they are not consistent, the answer is 'NO'.

Now consider VW. The actual world must be in all spheres, or equivalently in the smallest one (nested property of Definition 4). So we have to add a consistency test between the first computed core (characteristic formulae of the smallest sphere) and classical formulae. If this test shows that these two sets are not consistent, the answer is 'NO'.

For the system VC, the modification is a little more important. The smallest sphere only contains a world: the actual one. So we have to modify the method to compute the first core. This particular core initially contains calssical formulae. The positive conditionals are treated as previously, but the negative conditionals must be treated in some different way: if $\neg(\phi\Rightarrow\psi)$ is a conjunct, then $\phi\wedge\psi$ or $\neg\phi$ is in the first core. If this core is not consistent, the answer is 'NO'.

3.4 A TABLEAU LIKE PRESENTATION, NP-EXAMPLE.

We are going to prove that $((\alpha\Rightarrow\beta)\wedge(\beta\Rightarrow\gamma)\wedge(\alpha\Rightarrow\neg\gamma)\wedge\neg(\alpha\Rightarrow\neg\alpha))\rightarrow\alpha$ is not a theorem of NP. The normal disjunctive form of the negation of this formula is: $(\alpha\Rightarrow\beta)\wedge(\beta\Rightarrow\gamma)\wedge(\alpha\Rightarrow\neg\gamma)\wedge\neg(\alpha\Rightarrow\neg\alpha)\wedge\neg\alpha$. So we will look for a model of this last formula. The method presented here is exactly the one used in the V-Procedure modified as explained in section 3.3. Only the form differs.

First, given a set Γ of classical literals and conditionals, we write all these formulae under a double line ordering them in the following way: first the positive conditionals ($\phi\Rightarrow\psi$), secondly the negative ones $\neg(\phi\Rightarrow\psi)$, and finally the classical formulae.

$\alpha\Rightarrow\beta$	$\beta\Rightarrow\gamma$	$\alpha\Rightarrow\neg\gamma$	$\neg(\alpha\Rightarrow\neg\alpha)$	$\neg\alpha$

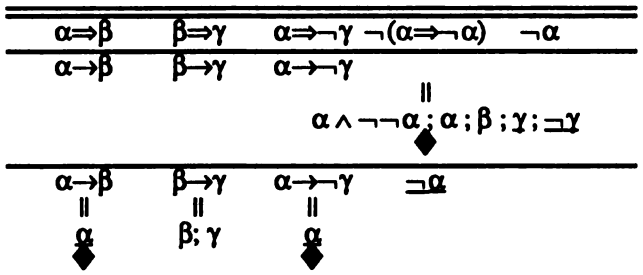
To compute the core of this set, we begin to write a simple line. Under each positive conditional, we rewrite it, replacing the main conditional operator ($\Rightarrow$) by a material implication.

$\alpha\Rightarrow\beta$	$\beta\Rightarrow\gamma$	$\alpha\Rightarrow\neg\gamma$	$\neg(\alpha\Rightarrow\neg\alpha)$	$\neg\alpha$
$\alpha\rightarrow\beta$	$\beta\rightarrow\gamma$	$\alpha\rightarrow\neg\gamma$		

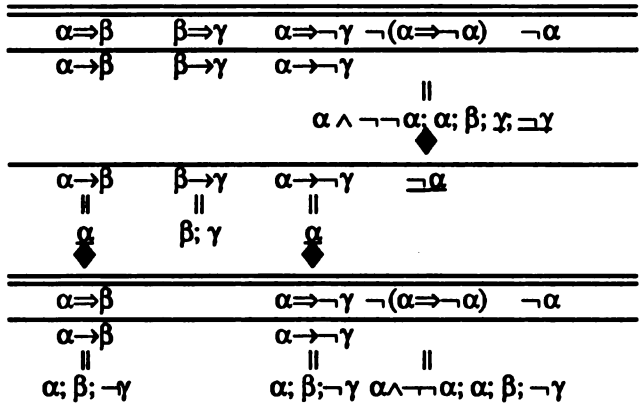
But this set is not the core in its whole: negative conditionals have not been considered yet. Given a negative conditional $\neg(\phi\Rightarrow\psi)$, we have to test if $\phi\wedge\neg\psi$ is consistent with the (temporary) core. If an inconsistency appears, the symbol ◆ is used. This is done for one negative conditional. If it is consistent, another one is tested until all of them are considered. If $\phi\wedge\neg\psi$ is not consistent with the core, we add $\neg\phi$ to the core. But before, we put another simple line to note that there is a modification at this step, and we write under it all the formulae known to belong to the core. When all negative conditionals have been considered, the core is totally computed.

$\alpha\Rightarrow\beta$	$\beta\Rightarrow\gamma$	$\alpha\Rightarrow\neg\gamma$	$\neg(\alpha\Rightarrow\neg\alpha)$	$\neg\alpha$
$\alpha\rightarrow\beta$	$\beta\rightarrow\gamma$	$\alpha\rightarrow\neg\gamma$		
		$\Vert$		
		$\alpha\wedge\neg\neg\alpha$; α ; β ; γ ; $\sqsupset\gamma$		
		◆		
$\alpha\rightarrow\beta$	$\beta\rightarrow\gamma$	$\alpha\rightarrow\neg\gamma$	$\neg\alpha$	

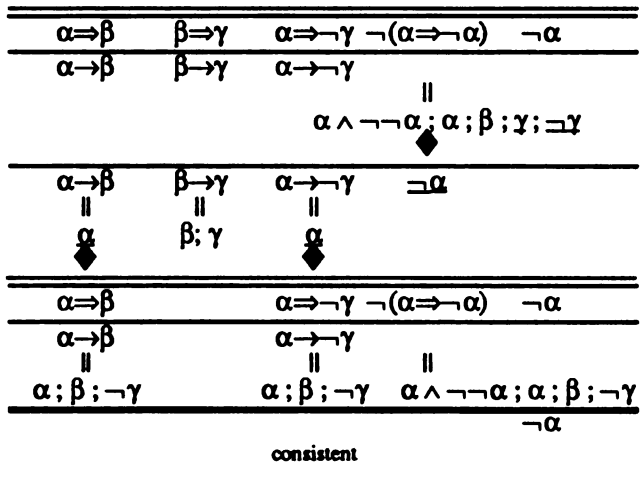
Remember that captured conditionals are the ones whose hypothesis are consistent with the core. Clearly, negative conditionals are captured iff they do not have the symbol ◆ under them. So only positive conditionals have to be tested. Under such a conditional, we write its premise to test consistency with the core. When this is done we can say that the conditionals without a ◆ under them are captured.



To compute the just including sphere, only uncaptured conditionals are considered. They are rewritten under a new double line, the new core is computed, and so on. These operations must be repeated until all conditionals are captured, or none can be captured any more.



Remember that if there were some uncaptured negative conditional, no model could be found. Here we have to go on. Since we are interested in conditional logic NP, the consistency of classical formulae with the core of uncaptured conditionals must be checked. This is done under a large single line. Here, since there is no uncaptured conditional, we simply put under the large line all the classical formulae and the final "tableau" is:



There is no failure here, so some model can be found for this formula. In fact, it is not difficult to extract. Each double line corresponds to a new sphere. Let us build it by creating a world, for each conditional without ◆ under it. We obtain:

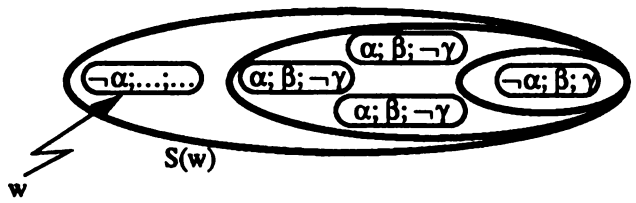


Figure 4: Coutermodel

This NP-model, which can be simplified, satisfies the negation of $((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \gamma) \wedge (\alpha \Rightarrow \neg\gamma) \wedge \neg(\alpha \Rightarrow \neg\alpha)) \rightarrow \alpha$. So we can conclude that the formula is not a theorem.

3.5 COMPARISON WITH OTHER APPROACHES.

Some correspondences have been checked or proved between couterfactual logics and modal logics [Delgrande 87] [Boutilier 90] [Lamarre 91] such that it is possible to use modal methods to prove theorems of conditional logics. The semantics of modal logic S4.3 is very close to the sphere semantics presented here. In fact, the conditional used here can be defined in terms of modal operators: $\alpha \Rightarrow \beta =_{Def} \Box(\alpha \rightarrow \Diamond(\alpha \wedge \Box(\alpha \rightarrow \beta)))$, and using this definition, some correspondences exist. If only formulae without nested conditional operators are considered then $\vdash_{NP} \alpha$ iff $\vdash_{S4.3} \alpha$[5]. If only Boolean combinations of flat conditionals are considered, then $\vdash_{VN} \alpha$ iff $\vdash_{S4.3} \alpha$ iff $\vdash_{NP} \alpha$. If we were only interested in such formulae, it would be possible to use an S4.3 theorem prover such as a tableau method [Zeman 73], [Hughes 84], or a sequent system [Goré 92]. But this approach is limitted: very few logics can be considered, the type of formulae is restricted, and in addition the translation is a bit complex.

But some theorem provers for conditional logics have been studied. Groeneboer and Delgrande [Groeneboer 88] have presented a theorem prover for some normal conditional logics, but their method is based on the correspondence with Kripke models for modal logics. Indeed they first compute an S4-model (a tree) and complete the accessibility relation to make it forward connected (if w_1Rw_2 and w_1Rw_3 then w_2Rw_3 or w_3Rw_2), in order to obtain an S4.3 model. This method is similar to the one proprosed in [Hughes 68] to obtain an S4.3 theorem prover. So, the same comments can be done.

4 MONOTONIC PROPERTIES OF NON-MONOTONIC INFERENCE RELATIONS

The study of non-monotonic properties of common-sense knowledge and reasoning is a main problem in artificial intelligence. Indeed, because of these properties it is not possible to capture such notions only using classical

[5] To consider full language, nested axioms must be added to NP, see [Lamarre 91].

(monotonic) logics. So different formalisms have been proposed to deal with this problem (see [Leasombe 89] for a survey). More recently, non-monotonic inference relations have been studied considering their properties. For example Kraus and al. [Kraus 90] have defined a class of non-monotonic inference relations. Rational inference relations have been defined in these terms. Lehmann and al. [Lehmann 89] have proved a representation theorem for these relations which gives some semantics based on preference relations in Shoham style [Shoham 87]. It has been shown, using semantics, that this definition is close to conditional logics [Kraus 90] [Nejdl 91] [Bell 90] [Katsuno 91], and more recently, using a systematic translation, that system V exactly corresponds to this definition [Crocco 92]. This makes us able to claim:

Theorem 2 *A theorem prover for monotonic properties of non-monotonicity*

Let α be a conditional formula obtained by translation[6] of a non-monotonic property.

$\vdash_{Proc,V} \alpha$ iff the non-monotonic property associated to α is verified by all rational inference relations.

So, the procedure presented in this paper is a tool to study rational inference relations via their common properties.

5 A NON-MONOTONIC PROVER

But this theorem prover can also be modified in order to have a non-monotonic behavior. This could be seen as a definition of a non-monotonic system based on conditional logic as in [Nute ??], [Delgrande 88], ... Let us consider the following particular model:

Definition 9 *Big Normal Model*

Let Γ be a set of conditionals without nested conditional operators and classical formulae.

Let us consider the following definition:

$C_0 = \Gamma$.

$s_i = \{w: w \text{ is a classical interpretation, and } w \models Core(C_i)\}$

$C_{i+1} = C_i \setminus \{\varphi : \varphi \text{ is a conditional captured by } s_i\}$

The model $M = \langle \mathcal{W}, S, v \rangle$ is defined by:

$\mathcal{W} = \bigcup_i s_i$

$S(w) = \{ s_i : s_i \text{ is defined as previously} \}$ for any w of $\mathcal{W}$.

v is defined as expected.

This corresponds to the particular model computed by the procedure Consistent. So, if Γ is consistent, there is a world w such that $M,w \models \Gamma$. The particularities of this model is that it contains as many worlds[7] as possible and that each world is in a sphere as little as possible. In some other terms, each sphere, beginning by the smallest one, is as big as possible. Now we can define a non-monotonic inference relation as follows:

Definition 10 *Non-monotonic inference relation: Normal inference relation.*

Let Γ be a set of flat conditionals and classical formulae.

Let $M = \langle \mathcal{W}, S, v \rangle$ be the Big Normal Model of Γ.

Let α be a formula.

$\Gamma \vdash_V \alpha$ [8] iff $M,w \models \alpha$ for any world w of $\mathcal{W}$ such that $M,w \models \Gamma$[9].

The procedure may be modified in order to implement this definition. It is not very difficult to see that $\vdash_V \alpha$ iff $\vdash_V \alpha_i$ for some i such that α_i is a disjunct of the normal disjunctive form of α, as for monotonic inference relation. Building the Big Normal Model of Γ sphere by sphere aspreviously done in the procedure V-Consistent, at each step we have to test if conditionals of α_i are captured by the sphere, and if it is the case, if they are satisfied in it. Since in logic V there is no link between conditional and classical formulae, the classical formulae of α_i will only be checked with classical formulae of Γ: "do classical formulae of Γ entails classical formulae of α_i?".

Example 1

We are going to prove that: $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \chi) \vdash_V (\alpha \Rightarrow \chi)$. Intuitively, this can be the translation of the non-monotonic rules: "Penguins are birds", "Birds have bills", "Penguins have bills".

	$(\alpha \Rightarrow \beta)$	$(\beta \Rightarrow \chi)$	$(\alpha \Rightarrow \chi)$
Core = {	$(\alpha \rightarrow \beta)$	$(\beta \rightarrow \chi)$ }	
	$\|\|$	$\|\|$	$\|\|$
	$\alpha; \beta; \chi$	$\beta; \chi$	$\alpha; \beta; \chi$
			Captured
			$Core \vdash (\alpha \rightarrow \chi)$
			is a valid classical inference

The left part exactly corresponds to the procedure used to compute a model for $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \chi)$. The new part is the right one, and it corresponds to the non-monotonic behavior. Indeed, conditionals of the right part are not considered to compute the core, but we check if they are captured, and if they are, we check if they are satisfied. In this example, we can conclude that the non-monotonic entailment is verified since the conditional is captured and satisfied.

Example 2

We are going to prove that $(\alpha \Rightarrow \beta) \wedge (\alpha \Rightarrow \neg\delta) \wedge (\beta \Rightarrow \chi) \wedge (\beta \Rightarrow \delta) \vdash_V \alpha \Rightarrow \chi$ is not a correct non-monotonic inference. Intuitively, this can be the translation of the non-monotonic

[6] See [Crocco 92].

[7] Here we don't make any difference between worlds and classical interpretations, this is possible since there is no nested conditional operator.

[8] Here the non-monotonic inference relation is defined on V models as mentioned by subscript letter, but it can also be defined using other conditional logics.

[9] This quantification is not necessary if classical formulae are not considered.

rules: "Penguins are birds", "Penguins don't fly", "Birds have bills", "Birds fly", "Penguins have bills".

$\alpha\Rightarrow\beta$	$\alpha\Rightarrow\neg\delta$	$\beta\Rightarrow\chi$	$\beta\Rightarrow\delta$	$\alpha\Rightarrow\chi$
Core={$\alpha\rightarrow\beta$	$\alpha\rightarrow\neg\delta$	$\beta\rightarrow\chi$	$\beta\rightarrow\delta$}	
‖	‖	‖	‖	‖
$\alpha;\beta;\sqsupset\underline{\delta};\underline{\delta}$	idem	$\beta;\chi;\delta$	$\beta;\chi;\delta$	$\alpha;\beta;\sqsupset\underline{\delta};\underline{\delta}$
◆	◆			Not captured ◆

$\alpha\Rightarrow\beta$	$\alpha\Rightarrow\neg\delta$	$\alpha\Rightarrow\chi$
Core={$\alpha\rightarrow\beta$	$\alpha\rightarrow\neg\delta$}	
‖	‖	‖
$\alpha;\beta;\neg\delta$	$\alpha;\beta;\neg\delta$	$\alpha;\beta;\neg\delta$
		Captured
		Core $\vdash(\alpha\rightarrow\chi)$
		is not a valid classical inference
		◆

In the first sphere, we don't have to test if $(\alpha\rightarrow\chi)$ is classically entailed by the core since the conditional is not captured.

This result is less expected than the previous one. Indeed, the fact that Penguins do not fly does not seem relevant with their bills. This shows that this non-monotonic inference relation is not completely correct according to our intuition. But it does not mean that it is not an interesting one. For example, it can be linked to Pearl's system Z [Pearl 90], which from a natural ordering (Z-ordering) on interpretations (in Shoham style [Shoham 87]), on default rules (positive conditionals), and on consistent classical formulae, defines a non-monotonic inference relation noted $\vdash_{\overline{1}}$.

Theorem 3 *Correspondence with Pearl's System Z.*
Let Γ be an ε-consistent[10] set of positive conditionals without nested conditional operators.
Let α and β be classical formulae such that $\alpha\wedge\beta$ and $\alpha\wedge\neg\beta$ are consistent.
$\alpha\vdash_{\overline{1}}\beta$ with respect to Γ iff $\Gamma\vdash_{V}(\alpha\Rightarrow\beta)$

This approach can also be used to generalize Pearl's system Z in different ways. Considering the procedure, any V-consistent set of conditionals can be used. $\alpha\wedge\beta$ and $\alpha\wedge\neg\beta$ have no longer to be consistent (and not only ε-consistent sets). The language, and then the expressive power can be augmented in order to deal with negative conditionals, nested conditionals and also classical formulae. Indeed, the procedure deals with any kind of formulae such that we don't have to restrict the language to positive conditionals. The main interest of these possible modifications, is to obtain a formalism with a greater expressivity. Another main interest is maybe that the basis of the non-monotonic inference relation may be changed. Indeed, the inference relation presented above is based on the conditional logic V, but VN, NP, VW, VC, ... may also be used since the theorem prover can be adapted to other semantics using the sphere notion.

[10] ε-consistency is more restrictive than V-consistency. See [Pearl 89,90] for more details. $\vdash_{\overline{1}}$ is defined under ε-consistency restriction.

6 CONCLUSION AND FUTURE WORK

This paper has presented a method to theorem proving for the class of conditional logics using sphere semantics. As these logics can be used to represent a large and useful class of notions such that revision, normality, possibility, our theorem prover has a wide range of applications. Moreover it can be expressed in a tableau style.

In a second part, we have also shown how this theorem prover can be used in the non-monotonic area. Indeed, the theorem prover is a tool to study common properties of non-rational inference relation. And, with some little modifications it defines, and implement, a particular non-monotonic inference relation.

This last point is not yet totally explored. Indeed some other possibilities exist to adapt this theorem prover. For example, it would be possible to associate a number to each conditional. And at each sphere computation, instead of abandoning all captured conditionals, it would be possible to abandon only those, among captured ones, the associated number of which are minimal (or maximal depending on the meaning of this number). This could be a way to make non-monotonic deductions closer to the intuition. Maybe, a non-monotonic system defined in this way would be very close to the system Z+ [Goldzmidt 91], or to possibility theory.

References

[Bell 90] Bell, J., "The Logic of Nonmonotonicity", Artificial Intelligence, vol. 41, pp. 365-374, 1990.

[Boutilier 90] Boutilier, G., "Conditional Logics of Normality as Modal Systems", in AAAI, P., MIT Press, 1990, pp. 594-599.

[Chellas 75] Chellas, B.F., "Basic Conditional Logics", Journal of Philosophical Logic, no. 4, pp. 133-153, 1975.

[Crocco 92] Crocco, G. and Lamarre, P., "On the Connection between Non Monotonic Inference Systems and Conditional Logics", KR'92, 1992.

[Delgrande 87] Delgrande, J.P., "A First Order Conditional Logic for Prototypical Properties", Artificial Intelligence, no. 33, pp. 105-130, 1987.

[Delgrande 88] Delgrande, J.P., "An approach to Default Reasoning Based on First Order Conditional Logic: Revised Report", Artificial Intelligence, no. 36, pp. 63-90, 1988.

[Delgrande 89] Delgrande, J.P., "More Songs About Buildings and Food-Collected Extended and Augmented Proofs for a Logic of Default Properties and for Default Reasoning I", Tech. Rep., Simon Fraser University, Burnaby, B.C., CANADA, CSS/LCCR TR 89-12, December 1989.

[Fariñas 91] Fariñas del Cerro, L. and Herzig, A., "A Modal Analysis of Possibility Theory", in Fundamentals of Artificial Intelligence Reasea.ch, International Workshop FAIR'91 (proceedings) Jorra.id, P. and Kelemen, J., Eds. Springer Verlag, Septembx r 199*, pp. 11-18.

[Goldzmidt 91] Gol 'zmidt, M. and Pearl, J., "System Z+: A for-

malism for reasoning with variable-strenght defaults", in AAAI , 1991.

[Goré 92] Goré, R., "Cut-free Tableau and Sequent Systems for Propositional Modal Logics S4.3, S4.3.1, S4.14", Forthcoming, 1992.

[Groeneboer 88] Groeneboer, C. and Delgrande, J.P., "Tableau-Based Theorem Proving in Normal Conditional Logics", in AAAI, 1988.

[Hughes 68] Hughes, G. and Cresswell, M., An Introduction to Modal Logic. Methuen Londres, 1968.

[Hughes 84] Hughes, G. and Cresswell, M., A companion to Modal Logic. Methuen Londres, 1984.

[Katsuno 91] Katsuno, H. and Satoh, K., "A Unified View of Consequence Relation, Belief Revision and Consequence Logic.", in Proceedings of the twelfth International Join Conference on Artificial Intelligence (I.J.C.A.I.)., vol. 1, Morgan Kaufmann, August 1991, pp. 406-412.

[Kraus 90] Kraus, S., Lehmann, D., and Magidor, M., "Nonmonotonic Reasoning, Preferential Models and Cumulative Logics", Artificial Intelligence, no. 44, pp. 167-207, 1990.

[Lamarre 91] Lamarre, P., "S4 as the Conditional Logic of Nonmonotonicity", in Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, Allen, J.A., Fikes, R., and Sandewall, E., Morgan Kaufmann, San Mateo CA, 1991, pp. 357-367.

[Lamarre 92] Lamarre, P., "Raisonnements non-monotones et logiques monotones", Rapport de Thèse, IRIT, Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse Cedex, France, Forthcoming, 1992.

[LeaSombe 89] LéaSombé, Raisonnement sur des Informations Incomplètes en Intelligence Artificielle. Tekna, 1989.

[Lehmann 89] Lehmann, D., "What does a Conditional Knowledge Base Entails", KR'89, R.J. Brachman, H.J. Levesque, R.R., Morgan Kaufmann, 1989, pp. 212-222.

[Lewis 86] Lewis, D., Counterfactuals. Basil Blackwell Ltd, 1986 (first published 1973).

[Nejdl 91] Nejdl, W., "The P-Systems: A Systematic Classification of Logics of Non Monotonicity", in Proceedings Night National Conference on Artificial Intelligence (AAAI), vol. 1AAAI Press / The MIT Press, July 14-19 1991, pp. 366-372.

[Nute 80] Nute, D., Topics in Conditional Logics. D. Reidel Publishing Company, 1980.

[Nute 84] Nute, D., "Conditional Logic", in Handbook of Philosophical Logic, Gabbay, D. and Guenthener, F., Eds. D. Reidel Publishing Company, 1984, pp. 387-349.

[Nute ??] Nute, D., "A Non-Monotonic Logic Based on Conditional Logic", ACM Research Report 01-0007, ??.

[Pearl 89] Pearl, J., "Probabilistic Semantics for Nonmonotonic Reasoning: A Survey", KR'89, Brachman, R.J., Levesque, H.J., and Reiter, R., Morgan Kaufmann, 1989, pp. 505-516.

[Pearl 90] Pearl, J., "System Z: A Natural Ordering of Defaults with Tractable Applications to Nonmonotonic Reasoning", in Theoretical Aspects of Reasoning About Knowledge (TARK-III), Morgan Kaufmann, 1990, pp. 121-135.

[Shoham 87] Shoham, Y., "A semantical Approach to Nonmonotonic Logics (extended abstract)", in Proceedings Logics in Computer Science, 1987, pp. 275-279.

[Smullyan 68] Smullyan, First Order Logic. Springer Verlag, 1968.

[Stalnaker 68] Stalnaker, R.C., "A Theory of Conditional", in Ifs, Hamer, W.L., Stalnaker, R., and Pearce, G., Eds. Basil Blackwell Publisher, 1968, pp. 41-55.

[Zeman 73] Zeman, J.J., Modal Logics: The Lewis Modal Systems. Oxford University Press, 1973.

Bounding Introspection in Nonmonotonic Logic

Grigori Schwarz
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716

Abstract

We investigate the effect of restricting introspection in nonmonotonic modal logics to formulas of bounded modal depth. First we show that mechanically restricting introspection to objective sentences (which leads to the well-known notion of a "grounded nonmonotonic logic") produces systems with logically counterintuitive properties. Then we show that if the underlying monotonic modal logic contains the axiom schema $K\psi \supset KK\psi$, then full negative introspection contains some restricted form of positive introspection (which we call "belief introspection") and can be reduced to both negative knowledge introspection and positive belief introspection w.r.t. objective formulas. For a wide class of modal logics full negative introspection can be reduced to negative introspection w.r.t. objective formulas if the set of initial assumptions (axioms) does not contain nested modalities.

1 Introduction

In the knowledge representation theory there has been considerable interest in the reasoning of rational agents with self-reflection capabilities ([MD80, McD82, Moo85, Lak92] a.o.). It means that an an agent is capable of reasoning not only about the world, but also of reasoning about its own knowledge and belief about the world. It is interesting to understand to what extent it is essential to assume that a rational agent is capable of reasoning about knowledge about knowledge about knowledge ... asf. In other words, how natural (or essential) is setting a boundary on the depth of introspection?

This problem has been thoroughly investigated for the case of several reflecting agents. In this case agents are capable of reasoning not only about their own knowledge and belief, but also about knowledge and belief of other agents. It is well known that in the case of several agents it is impossible to restrict introspection by any finite depth (see, e.g., [HM90]).

However in the one-agent case the issue is not quite clear. Lakemayer [Lak92] argues that the epistemic state of an agent should be uniquely determined by the set of *objective beliefs* of the agent. Similar arguments can be given in favor of the opinion that *objective introspection* in the one-agent case should determine the higher depth introspection.

In the present paper we investigate to what extent the depth of introspection can be bounded in the modal nonmonotonic logics which were introduced by McDermott and Doyle [MD80, McD82] and Moore [Moo85] as formalizations of the one-agent introspective reasoning.

First recall the basic ideas of these approaches.

Let $\mathcal{L}$ be the usual propositional modal language, i.e the propositional language augmented by the modal necessity operator K. The intended interpretation of $K\psi$ is "ψ is known". The basic notion of McDermott and Doyle's logic is an expansion of a given axiom set. Let $\mathcal{S}$ be any monotonic logic in $\mathcal{L}$. By an *$\mathcal{S}$-expansion of A* we mean a set T of formulas such that

$$T = \{\varphi : A \cup \{\neg K\psi : \psi \notin T\} \vdash_{\mathcal{S}} \varphi\}. \qquad (1)$$

An expansion of a given formula set A can be understood as a candidate for a set of epistemic consequences of A. The term $\{\neg K\psi : \psi \notin T\}$ expresses the negative introspection of an agent: if a formula ψ is not in agent's belief set, the agent knows that it does not know ψ.

Note the presence of a *monotonic* modal logic as a subscript in (1). This is a great advantage of McDermott and Doyle's approach: by varying the underlying monotonic logic we can capture different aspects of reasoning about knowledge and belief. These notions are very complicated and rather vague (see, e.g. Lenzen [Len78] for a comprehensive survey), so we hardly can expect to get just one logic working well for all purposes. The paper [MST91] contains a com-

prehensive investigation of the dependence of the nonmonotonic logic on the underlying monotonic modal logic; however the problem of bounded introspection remains uninvestigated.

McDermott and Doyle's approach is rather powerful. If we interpret the modality K as "is provable", then it captures the notion of negation as failure [Rei78] (see also [Tru91b] for more information about the relationship of negation as failure to McDermott and Doyle's approach). If we interpret K as "is known", or "is believed", then it naturally captures introspective reasoning. It is also capable of capturing default logic [Tru91a]. Moore's autoepistemic logic is a special case of McDermott and Doyle's logic [Shv90].

We concentrate in this paper on the introspective aspects of the modal approach to nonmonotonic logic. The introspective aspects seem to be important. If we interpret A as a knowledge base, then the ability to express the relationship between the real world and the knowledge implicit in A increases the expressive power of the knowledge base. Lifschitz [Lif91] argued in favor of data bases admitting epistemic queries like "Does KB know, whether p holds?".

R. Moore [Moo85] argued that McDermoott and Doyle's equation (1) captures only negative introspection, and proposed the following modification to that equation:

$$T = \{\varphi : A \cup \{\neg K\psi : \psi \notin T\} \cup \{K\psi : \psi \in T\} \vdash_{\mathcal{S}} \varphi\}, \quad (2)$$

where $\mathcal{S}$ is the propositional calculus in the modal language. The term $\{K\psi : \psi \in T\}$ represents the positive introspection of an agent. Solutions to equation (2) with the propositional calculus as $\mathcal{S}$ are called *stable expansions of* A, and the logic based on the notion of a stable expansion is known as "autoepistemic logic".

Note that, in principle, we could also consider equation (2) with an arbitrary modal logic $\mathcal{S}$. Moore proved that the class of solutions to (2) does not change, if we consider $\mathcal{S}$ to be any modal logic contained in K45 (that is, S5 without the axiom schema $K\varphi \supset \varphi$) instead of propositional calculus. In [McD82] it was proved that S5 is a bad basis for nonmonotonic logic: nonmonotonic S5 collapses into the monotonic S5. (It was proved for the formulation (1), which does not contain positive introspection. By using that fact the same property can easily be obtained for equation (2) with $\mathcal{S}$ = S5.) It was also proved in [McD82] that if a logic $\mathcal{S}$ contains the necessitation rule (from φ infer $K\varphi$) then all $\mathcal{S}$-expansions are closed under S5. Each $\mathcal{S}$-expansion is trivially closed under $\mathcal{S}$. Segerberg [Seg71] described all the logics properly containing S5; they are, in a sense, degenerate, so it makes no sense to consider logics not contained in S5 as a basis for nonmonotonic logic.

By the *modal depth* $m(\varphi)$ of a formula φ we call the maximal depth of nesting the modal operator K in φ. More precisely, we define $m(\varphi)$ by recursion on the complexity of φ as follows: $m(p) = 0$ if p is a propositional variable; $m(\varphi \wedge \psi) = m(\varphi \vee \psi) = \max(m(\varphi), m(\psi)$; $m(\neg\varphi) = m(\varphi)$; $m(L\varphi) = m(\varphi)+1$. By $\mathcal{L}_n$ we denote the fragment of $\mathcal{L}$ consisting of formulas ψ with $m(\psi) \leq n$. Clearly, $\mathcal{L}_0$ is exactly the set of all objective (that is, not containing K) formulas.

Konolige [Kon88] proved that equation (2) is equivalent to the following one:

$$T = Cn_{\mathbf{K45}}(A \cup \{\neg K\varphi : \varphi \in \mathcal{L}_0 \setminus T\} \cup \{K\varphi : \varphi \in \mathcal{L}_0 \cap T\}).$$

In other words, if we take K45 as an underlying logic rather than just the propositional calculus, then full introspection can be reduced to introspection w.r.t. objective formulas only.

Moore's logic produces some expansions which are commonly considered as ungrounded. For example, the theory $\{Kp \supset p\}$ ("if p is known then p") has a stable expansion containing p, which seems to be quite counterintuitive. In order to get rid of ungrounded expansions, Konolige considered the equation (1) for K45 restricting negative introspection to the objective formulas only. Expansions obtained in this way are called *moderately grounded*. The same restriction of negative introspection to objective formulas for S5 was considered by Tiomkin and Kaminski [TK90, Kam91]. We examine this way of restricting introspection in Section 3 and argue that it is logically unsatisfactory: the notion of a grounded expansion is unstable w.r.t introduction of explicit definitions: adding formulas of the form $q \equiv \varphi$, where q is a new propositional variable, to an initial assumption set A, may affect the consequences of A not containing q.

In Section 4 we show that the unrestricted negative introspection given by (1) can be reduced to both negative and positive introspection w.r.t. objective formulas only. But the positive introspection of (2) needs to be modified: in place of $\{K\psi : \psi \in T\}$ we need to use $\{\neg K\neg K\psi : \psi \in T\}$.

This form of introspection admits a very natural epistemic interpretation. There has been much written, both in the philosophical and AI literature, about the relationship between knowledge and belief (see, e.g. [Hin62, Len78, Len79, MS89, Voo91]). The both notions are vague and can be understood in many different senses. We will consider so called rational belief. A *rational* agent believes A if it is convinced that A is true, and it is also convinced that it can justify the truth of A. We can say that the rational agent thinks that it knows A. Thus, the agent does not distinguish between its own knowledge and belief; it is an omniscient external observer who can distinguish between agent's real knowledge and just beliefs.

This understanding of rational belief may seem too strong. We could call it "very rational belief". Lenzen

in [Len78] introduced for such "very rational belief" the special modal operator C to distinguish between $B\varphi$ (the agent believes that φ) and $C\varphi$ (the agent is convinced that φ). We will refer to the epistemic notion corresponding to C as "(very) rational belief".

Lenzen [Len78] argued that the following correspondence between knowledge and rational belief is true:

$$C\psi \equiv \neg K \neg K \psi.$$

At first look, this correspondence seems to be unjustified, especially the implication from the right hand side to the left one. But if we rewrite it in the logically equivalent form $\neg C\psi \equiv K\neg K\psi$, then it becomes intuitively quite acceptable. Thus, our main result can be formulated as follows: **for a wide class of modal logics (namely, for all logics between K4 and S5), unbounded negative introspection may be reduced to objective negative introspection w.r.t. knowledge and objective positive introspection w.r.t. rational belief.**

Note that because all $\mathcal{S}$-expansions are stable, we have for each $\mathcal{S}$-expansion T of any A, $\{\neg K\neg K\psi : \psi \in T\} = \{\neg K\neg K\psi : \neg K\psi \notin T\}$, so positive belief introspection w.r.t. ψ formally corresponds to negative knowledge introspection w.r.t. "ψ is unknown" – this explains how in principle negative introspection can be reduced to positive introspection.

The reason why different authors ([Kon88, TK90, Kam91]) have tried to save as much as possible of S5 is probably that, at first look, all the modal axioms of S5 seem to be reasonable under the interpretation of K as "is known" (see, e.g., [HM85]). On the other hand, the desire to restrict negative knowledge introspection to introspection with respect to objective propositions only seems to be quite natural.

Lenzen [Len79] argues that S5 is inappropriate as a logic of knowledge, and suggests the logic S4.2 for that role. We show that for a wide class of logics which includes S4.2, S4 and many others (but does not include KD45), if the set of initial assumptions A itself does not contain nested modalities, then the full negative introspection in (1) can be reduced to just negative introspection w.r.t. objective formulas – that is, in this case all expansions are grounded in the sense of [Kon88, TK90].

2 Preliminaries

Monotonic modal logics under discussion contain modus ponens and the necessitation rule as the only inference rules, all instances of propositional tautologies in the full modal language, and some axiom schemata from the following list:

K: $K(\varphi \supset \psi) \supset (K\varphi \supset K\psi)$

4: $K\psi \supset KK\psi$

T: $K\psi \supset \psi$

5: $\neg K\psi \supset K\neg K\psi$.

K is the logic containing schema K only, T is K with schema T, S4 contains schemata K, T and 4, K4 contains schemata K and 4, K45 contains schemata K, 4 and 5. S5 contains all the schemata listed above.

By $\vdash_{\mathcal{S}}$ we denote the derivability relation in a logic $\mathcal{S}$ by using modus ponens and the necessitation rule. By $Cn_{\mathcal{S}}(A)$ we denote the set $\{\varphi : A \vdash_{\mathcal{S}} \varphi\}$. If $\mathcal{S}$ is the classical propositional logic in the modal language, we will skip $\mathcal{S}$ in the subscript.

We consider the possibility modality M as an abbreviation of $\neg K\neg$

Remark 2.1 To avoid possible confusion, let us note that in the literature, both in classical modal logic and in AI, the notation $A \vdash_{\mathcal{S}} \psi$ is used in different senses. In popular monographs [HC84, Che80] and in many papers it is used to denote that ψ is derivable from A and the *theorems of $\mathcal{S}$ by using modus ponens only*. We follow [Fey65, McD82, Moo85] and use this notation to denote that ψ is derivable from A *and the axioms of* $\mathcal{S}$ by means of both modus ponens and necessitation. Of course, this is just a notational difference, and all the results of our paper can be reformulated in terms of the alternative notation.

A set of formulas, T, is called *stable* if: T is closed under tautological consequences; for each formula $\psi \notin T$, $\neg K\psi \in T$, and for each formula $\psi \in T$, $K\psi \in T$. In other words, T is stable if it is deductively closed, and closed under both positive and negative introspection. Clearly, each $\mathcal{S}$-expansion of A is stable, but not vice versa: it does not follow from the fact that a set is closed under introspection rules that it can be obtained from the initial assumption set by means of these rules and axioms of $\mathcal{S}$. The notion of a stable set is the central one in the theory of introspective reasoning.

It was proved in [Moo85] that for any set A of objective formulas there is the unique stable set whose objective part coincides with the tautological closure of A. We will denote this unique stable set by $E(A)$. Moore [Moo84] proved that a set T is stable if and only if there is a Kripke model $\mathcal{M}$ with the universal accessibility relation, such that $T = \{\psi : \mathcal{M} \models \psi\}$. This fact implies that each stable set contains all instances of all axiom schemata of S5 (which was proved by McDermott [McD82]). McDermott [McD82] proved also that if $\mathcal{S}$ and $\mathcal{T}$ are modal logics containing the necessitation rule and $\mathcal{S} \subseteq \mathcal{T} \subseteq$ S5, then each $\mathcal{S}$-expansion is an $\mathcal{T}$-expansion as well. We will use the above facts throughout the paper without special references.

3 Grounded expansions and explicit definitions

Here we argue that the straightforward restriction of introspection to objective formulas leads to a notion which possess logically undesirable properties.

Several researchers (see [Kon88, TK90, Kam91, Tru91a]) have considered a restricted form of equation (1), which corresponds intuitively to restricting negative introspection to objective formulas only:

$$T = \{\varphi : A \cup \{\neg K\psi : \psi \in \mathcal{L}_0 \setminus T\} \vdash_{\mathcal{S}} \varphi\}. \quad (3)$$

Konolige [Kon88] considered the equation (3) for the case $\mathcal{S}$ = K45. Solutions to that equation are called *moderately grounded expansions of A*. The motivation for introducing this notion was the existence of K45-expansions (i.e. stable expansions) which Konolige considered intuitively ungrounded. The most typical example of an ungrounded expansion is the following. Consider the theory $\{Kp \supset p\}$. This theory has two stable expansions, $T_1 = E(\emptyset)$ and $T_2 = E(p)$. The latter one is considered in [Kon88] as ungrounded: the only reason to include p in the belief set is that p would be true if believed.

But this "ungroundness" seems to be superficial. Consider the theory $B = A \cup \{q \equiv Kp\}$. In other words, we introduce a new denotation, q for Lp. The introduction of a new denotation should not influence logical properties. And the extended theory has two stable expansions, $E(\neg q)$ and $E(p, q)$, which naturally correspond to T_1 and T_2 respectively. Both the expansions are moderately grounded. Thus, the introduction of a new definition may convert an ungrounded expansion into a grounded one. (Here we do not consider the strongly grounded expansions introduced in [Kon88], because we believe that that notion treats autoepistemic logic as a system of rules rather then as a *logic*.)

Nonmonotonic logic S5, investigated by McDermott [McD82], collapses to monotonic S5, hence is an unsatisfactory formalization of nonmonotonic reasoning. But on the other hand, all modal axioms of S5 seem to be natural if we interpret the modality K as "is known", and researchers have continued to use S5 as the basis for nonmonotonic logic, especially when interpreting modality K as "is known" (see e.g. [HM85]). Tiomkin and Kaminski [TK90, Kam91] investigated solutions to (3) for the logic S5. The solutions to (3) are called *ground* $\mathcal{S}$-expansions, and the corresponding nonmonotonic logic is referred to as *ground nonmonotonic* $\mathcal{S}$. Ground nonmonotonic S5 does not collapse into the monotonic S5, as (unground) nonmonotonic S5 does.

Let us consider the empty theory. It has $E(\emptyset)$ as the only ground S5-expansion. Now, let us introduce an explicit definition, $q \equiv Lp$. The theory $\{q \equiv Kp\}$ has two ground S5-expansions: $E(q)$ and $E(\neg q)$ (the latter is generated by the only expansion of $\emptyset$).

Thus, we have the following situation. We have no axioms, and have the unique knowledge set, in which nothing objective is known but tautologies, which is, of course, quite natural. Then we add an axiom, which is nothing but a new denotation for Kp. And we get two knowledge sets, in one of which p is known. So introducing a denotation makes it possible for an agent to know something objective. We consider this situation quite unnatural from the logical point of view. Introducing denotations must not lead to acquisition of new objective knowledge.

As we mentioned in the introduction, for nonmonotonic K45 (Moore's autoepistemic logic) a reduction of a similar nature has been provided by Konolige [Kon88]. Namely, he proved that T is a stable expansion of A if and only if

$$T = Cn_{\mathrm{K45}}(A \cup \{\neg K\varphi : \varphi \in \mathcal{L}_0 \setminus T\} \cup \{K\psi : \psi \in \mathcal{L}_0 \cap T\}).$$

The positive introspection is expressed here by the term $\{K\psi : \psi \in T \cap \mathcal{L}_0\}$. We can say that for nonmonotonic K45 full negative introspection can be reduced to both negative and positive *objective* introspection. This reduction fails for logics K, K4, S4 and many others, since we can obtain superfluous expansions (e.g. we obtain the "ungrounded" expansion $E(p)$ for the theory $\{Lp \supset p\}$).

4 Reduction to objective introspection

In this section we prove that if an underlying logic contains K4, then full negative introspection can be reduced to negative and modified positive introspection w.r.t. objective formulas only, but that fails, if the logic does not contain the schema 4.

The following technical lemmas imply the desired reduction almost immediately.

Lemma 4.1 *If T is stable and consistent then for each $\eta \notin T$,*
$\{\neg K\varphi : \varphi \in \mathcal{L}_0 \setminus T\} \cup \{\neg K\neg K\psi : \psi \in \mathcal{L}_0 \cap T\} \vdash_{\mathrm{K4}} \neg K\eta$.

Proof. See Appendix. □

Lemma 4.2 *Assume that $\mathcal{S}$ is any logic between* K *and* S5, *T is consistent and satisfies the equation*

$$T = Cn_{\mathcal{S}}(A \cup \{\neg K\varphi : \varphi \in \mathcal{L}_0 \setminus T\} \cup \{\neg K\neg K\psi : \psi \in \mathcal{L}_0 \cap T\}).$$

Then T is stable.

Proof. See Appendix. □

Proposition 4.3 *Let S be any normal modal logic contained in* S5 *and containing* K4. *Then T is an $\mathcal{S}$-expansion of A if and only if T satisfies the following equation:*

$$T = Cn_{\mathcal{S}}(A \cup \{\neg K\varphi : \phi \in \mathcal{L}_0 \setminus T\} \cup \{MK\psi : \psi \in T_0\}). \quad (4)$$

Proof. If T is an $\mathcal{S}$-expansion of A then (4) follows immediately from Lemma 4.1 and the fact that $\psi \in T$ implies $\neg K\psi \notin T$.

The converse implication follows from Lemma 4.2. □.

It was proved in [Shv90] that stable expansions are exactly K45-expansions. But we have $\vdash_{\mathbf{K45}} MK\psi \equiv K\psi$, hence the reduction given by Proposition 4.3 is equivalent to

$$T = Cn_{\mathcal{S}}(A \cup \{\neg K\varphi : \phi \in \mathcal{L}_0 \setminus T\} \cup \{K\psi : \psi \in T_0\})$$

which is exactly the reduction to objective introspection provided by Konolige [Kon88].

Schema 4 is essential for the Proposition 4.3. Without Schema 4 we cannot, in general, restrict the introspection in (1) to any bounded depth of nesting L. Let us call T an ***n-bounded $\mathcal{S}$-expansion of A***, if T satisfies the equation

$$T = Cn_{\mathcal{S}}(A \cup \{\neg K\phi : m(\varphi) \leq n,\ \varphi \notin T\}).$$

Proposition 4.3 implies, in particular, that if $\mathcal{S}$ contains K4, then each $\mathcal{S}$-expansion is a 1-bounded expansion. We show that axiom 4 is essential for this reduction.

Proposition 4.4 *For each n, there is a theory A such that A has a* K*-expansion (hence an* T*-expansion) which is not an n-bounded* T*-expansion of A (hence not an n-bounded* K*-expansion of A).*

Sketch of the proof. By $K^n\varphi$ we denote formula obtained by adding n K-s in front of φ.

Consider the theory $A = MK^{n+1}p \supset p$. Clearly $T = E(p)$ is the only K-expansion of A and the only T-expansion of A. Using the standard Kripke models technique, it is not hard to prove that

$$p \notin Cn_{\mathrm{T}}(A \cup \{\neg K\phi : m(\varphi) \leq n,\ \varphi \notin T\}).$$

The full proof is presented in the Appendix. □

5 Groundedness for theories without nested modalities.

The previous results show that if a logic contains Schema 4, then all expansions are 1-bounded, and for normal modal logics not containing 4 we cannot, in general, bound the depth of 1 negative introspection by any natural number. But the reader may note that our counterexample $ML^{n+1}p \supset p$ does not seem to be very natural. If we are interested in considering bounded introspection, why should we admit unbounded nesting of modalities in the axioms? We show in this section that for a wide class of modal logics, if we restrict the depth of nested modalities in axioms by n, then all expansions will be $(n-1)$-bounded. As a consequence we obtain that if A does not have nested modalities, then each expansion is a 0-bounded expansion, i.e. a grounded expansion ([TK90, Kam91, Tru91b]). This result seems to be important. When formalizing common sense reasoning patterns, we usually obtain formulas without nested modalities (see e.g. [Gel89, Mor89]). In such cases restricting to objective negative introspection maintains the power of the unrestricted negative introspection.

We need to recall some results of [Shv90, MST91]. For a large class of modal logics, a complete characterization of $\mathcal{S}$-expansions was given there.

By A^K we denote the set of all subformulas of A which begin with K. Let T be a consistent $\mathcal{S}$-expansion of A. Then T is stable. Let $\Psi = A^K \cap T$, $\Phi = A^K \setminus \Psi$. If $K\psi \in \Psi$ then $\psi \in T$, and if $K\psi \in \Phi$ then $\psi \notin T$. T is closed under propositional consequences, hence $S = Cn(A \cup \{\neg K\varphi : \varphi \in \Phi\} \cup \Psi \cup \{\psi : K\psi \in \Psi\}) \subseteq T$. We denote the unique stable set whose objective part is S_0 by $W_{A,\Phi}$. It was proved in [Shv90] that for many modal logics $\mathcal{S}$, any $\mathcal{S}$-expansion of A is equal to some $W_{A,\Phi}$. Namely, Φ is said to be $\mathcal{S}$-admissible for A, if $A \cup \neg\Phi$ is consistent with $\mathcal{S}$, and for each $K\psi \in \Psi$, $A \cup \neg\Phi \vdash_{\mathcal{S}} \psi$. It was shown in [Shv90] that for any logic $\mathcal{S}$ containing the necessitation rule and contained in S5, if $\Phi \subseteq A^K$ is $\mathcal{S}$-admissible for A, then $W_{A,\Phi}$ is an $\mathcal{S}$-expansion of A. In addition, for a wide class of modal logics, *each* $\mathcal{S}$-expansion is $W_{A,\Phi}$ for some $\mathcal{S}$-admissible for A set Φ. We will say that a modal logic $\mathcal{S}$ is *canonically characterized* iff for each A, each $\mathcal{S}$-expansion of A equals $W_{A,\Phi}$ for some $\mathcal{S}$-admissible for A set Φ. In [MST91] a sufficient condition for a logic to be canonically characterized is given. Almost all well-known modal logics (contained in S5) are canonically characterized: K, T, K4, S4, S4.2 and many others. Logic S4F, which has recently found interesting epistemic interpretations both in its monotonic and nonmonotonic variants (see [Tru91a, Voo91, ST92]) is canonically characterized too. This class is defined in terms of some elementary properties of Kripke models for $\mathcal{S}$.

Among "usual" normal logics, contained in S5, only S5, K45, KD45 and S4.4 are not canonically characterized.

We need the following auxiliary lemma which is analogous to Lemma 4.1.

Lemma 5.1 *If T is consistent and stable, then for each $\eta \in T$,*

$$(T \cap \mathcal{L}_0) \cup \{\neg K\psi : \psi \in \mathcal{L}_0 \setminus T\} \vdash_{\mathrm{K}} \eta.$$

Proof is presented in Appendix. □

Proposition 5.2 *Let $\mathcal{S}$ be any canonically characterized normal modal logic. Let $m(\varphi) \leq n+1$ for each formula $\varphi \in A$. Then each $\mathcal{S}$-expansion of A is n-bounded.*

Proof. Assume that T is an $\mathcal{S}$-expansion of A. Then for some $\mathcal{S}$-admissible for A set Φ, $T_0 = (Cn_{\mathcal{S}}(A \cup \{\neg K\phi : K\phi \in \Phi\}))_0$. Hence and from Lemma 5.1, using the fact that stable sets are closed under S5, we obtain

$$T = Cn_{\mathcal{S}}(A \cup \{\neg K\psi : K\psi \in \Phi\} \cup \{\neg K\psi : \psi \in \mathcal{L}_0 \setminus T\}).$$

But if $K\psi \in \Phi$, then $m(\psi) \leq n$. □

Corollary 5.3 *If logic $\mathcal{S}$ is canonically characterized and A does not contain nesting modalities, then each $\mathcal{S}$-expansion of A is a ground (i.e.0-bounded) $\mathcal{S}$-expansion of A.* □

The latter corollary applies to a wide class of modal logics: K, T, K4, S4, S4F and many others. It does not hold for S5 and K45, which is demonstrated by Konolige's examples of ungrounded expansions.

Thus, for canonically characterized modal logics, if a set of initial assumptions does not contain nested modalities, then we get a reduction even stronger than the reduction given by Theorem 4.3. Namely, in this case full negative introspection can be reduced to objective negative introspection. That is, all expansions are grounded in the sense of [Kam91].

6 Conclusions

We have examined the effect of restricting introspection to the formulas of low modality depth in the framework of McDermott and Doyle's modal approach to the formalization of nonmonotonic reasoning. It turns out that those effects depend on the monotonic modal logic which serves as the basis for the nonmonotonic logic.

We considered first the most popular nonmonotonic modal logics, namely Moore's autoepistemic logic and S5. We showed that the formal restriction of negative introspection to objective formulas leads to logics with a property we consider undesirable: introduction of explicit definitions may result in essential changes in the sets of nonmonotonic consequences. On the other hand, it is known that in these cases restricting to the negative introspection w.r.t. formulas without nested modalities, or restricting to both positive and negative introspection w.r.t. objective formulas maintains the whole power of unrestricted introspection. But nonmonotonic S5 with unrestricted introspection is known to collapse into monotonic S5, and unrestricted nonmonotonic K45 provides the so called ungrounded expansions which are commonly considered rather counterintuitive.

Then we considered other logics. The most important technical result we obtained is that, for any normal modal logic containing axiom schema $(K\psi \supset KK\psi)$ (and contained in S5), restricting to negative introspection w.r.t. formulas without nested modalities maintains the power of full introspection. Negative introspection w.r.t. the formula $\neg K\psi$ may be considered as the positive introspection w.r.t. ψ: for a stable set T, $\psi \notin T$ if and only if $\neg K\psi \in T$. If we express positive introspection in a belief set T as $\{MK\psi : \psi \in T\}$, then we can reduce unrestricted negative introspection to both negative and positive introspection w.r.t. *objective* formulas only.

This result admits a natural epistemic interpretation. The modality $MK\psi$ can be interpreted as "ψ is rationally believed" (or "the agent is convinced that ψ"), which seems to be intuitively justified. Then we can say that the unbounded negative introspection can be reduced to the negative knowledge introspection and positive belief introspection with respect to objective formulas.

For weaker logics, which do not contain schema $K\psi \supset KK\psi$, the reduction to introspection bounded by any depth of nesting modalities is in general impossible, which we have demonstrated by an example.

For a class of theories especially important for applications, those without nested modalities, for a wide class of modal logics including such well-known and important ones as K, T, K4 and S4 (but not S5 or K45!), we have an even stronger reduction: full negative introspection can be reduced to negative introspection for objective formulas.

7 Appendix. Proofs

In this section we present technical proofs omitted in the main body of the paper.

First we recall some elementary facts about derivability in modal logics containing K and 4.

Proposition 7.1 *(i)* $\vdash_{\mathbf{K}} M\varphi \wedge K\psi \supset M(\varphi \wedge \psi)$;
(ii) $\vdash_{\mathbf{K}} K(\varphi \supset \psi) \supset (M\varphi \supset M\psi)$;
(iii) $MK\varphi, MK\psi \vdash_{\mathbf{K4}} MK(\varphi \wedge \psi)$;
(iv) $M\varphi, MK\psi \vdash_{\mathbf{K4}} M(\varphi \wedge K\psi)$.

Proof. Parts (i) and (ii) are obvious and very well known. We sketch the proofs of (iii) and (iv).

(iii). From premises by using necessitation we obtain $KMK\varphi$, using (ii) and the schema 4 we get $MKK\psi$. Hence by (i), $M(MK\varphi \wedge KK\psi)$. Again by (i), (ii) and necessitation, $MM(K\varphi \wedge K\psi)$, hence using the theorem of K $K\varphi \wedge K\psi \supset K(\varphi \wedge \psi)$ and the dual form of schema 4 $(MM\eta \supset M\eta)$ we obtain $MK(\varphi \wedge \psi)$.

(iv). We obtain from premises, by Nec and 4, $KM\varphi \wedge MKK\psi$, hence by (i) $M(M\varphi \wedge KK\psi)$, again by (i), $MM(\varphi \wedge K\psi)$, hence by 4, $M(\varphi \wedge K\psi)$. □

Lemma 4.1 *If T is stable and consistent, then for each $\eta \notin T$,*

$$\{\neg K\varphi : \varphi \in \mathcal{L}_0 \setminus T\} \cup \{MK\psi : \psi \in \mathcal{L}_0 \cap T\} \vdash_{\mathbf{K4}} \neg K\eta. \quad (5)$$

Proof. We prove by induction on the depth of nesting of K in η that if $\eta \notin T$ then (5) holds, and if $\eta \in T$, then

$$\{\neg K\varphi : \varphi \in \mathcal{L}_0 \setminus T\} \cup \{MK\psi : \psi \in \mathcal{L}_0 \cap T\} \vdash_{\mathbf{K4}} MK\eta. \quad (6)$$

Assume that (5) holds for each $\eta \notin T$ with $m(\eta) < n$, and (6) holds for each $\eta \in T$ with $m(\eta) < n$.

First, we assume that $m(\eta) = n$ and $\eta \notin T$ and prove (5). We may assume without loss of generality that η is in conjunctive normal form, $\eta = \varphi_1 \wedge \ldots \wedge \varphi_m$, where each φ_i has a modal depth not larger than n and is of the form

$$\psi \vee K\eta_1 \vee \ldots \vee K\eta_k \vee \neg K\zeta_1 \vee \ldots \vee \neg K\zeta_l \quad (7)$$

where ψ is objective and that 1for each i, j $m(\eta_i) < n$, $m(\zeta_j) < n$. Clearly, some φ_i is not in T. Assume that such φ_i is of the form (7). Clearly each disjunct in (7) is not in T. T is stable, hence

$$\psi \notin T,\ \eta_i \notin T,\ \zeta_j \in T$$

for each i, j.

Denote $\{K\varphi : \varphi \in \mathcal{L}_0 \setminus T\} \cup \{MK\psi : \psi \in \mathcal{L}_0 \cap T\}$ by B. By the induction hypothesis and the definition of B we have

$$B \vdash_{\mathbf{K4}} \neg K\psi,\ B \vdash \neg K\eta_i,\ B \vdash MK\zeta_j$$

for each i, j. Rewriting $\neg K$ as $M\neg$ and applying (iv) of Proposition 7.1 several times, we obtain

$$B \vdash_{\mathbf{K4}} M(\neg\psi \wedge K\zeta_1 \wedge \ldots \wedge K\zeta_l). \quad (8)$$

By necessitation we have for each i, $B \vdash_{\mathbf{K4}} K\neg K\eta_i$. From this fact and (8) applying (i) of Proposition 7.1 several times, we obtain

$$B \vdash_{\mathbf{K4}} M(\neg\psi \wedge \neg K\eta_1 \wedge \ldots \wedge \neg K\eta_k \wedge K\zeta_1 \wedge \ldots \wedge K\zeta_l)$$

which may be rewritten as

$$B \vdash_{\mathbf{K4}} \neg K(\psi \vee K\eta_1 \vee \ldots \vee K\eta_k \vee \neg K\zeta_1 \vee \ldots \vee \neg L\zeta_l),$$

which is just $B \vdash_{\mathbf{K4}} \neg K\varphi_i$ which implies $B \vdash_{\mathbf{K4}} \neg K\eta$.

Now we assume that $\eta \in T$ and $m(\eta) = n$ and prove (6). Again, we can assume without the of generality that η is in a conjunctive normal form, $\eta = \varphi_1 \wedge \ldots \wedge \varphi_m$, and that each φ_i is of the form (7). Clearly each φ_i is in T. By Proposition 7.1, (iii), it is sufficient to prove that for each φ_i, $B \vdash_{\mathbf{K4}} MK\varphi_i$.

Assume that φ_i has the form (7). If for some j, $\eta_j \in T$, then we have, by the induction hypothesis, $B \vdash_{\mathbf{K4}} MK\eta_j$, but $K\eta_j \supset \phi_i$ is a theorem of K4. Applying the necessitation rule, schemata 4 and K and (ii) of Proposition 7.1 we easily obtain $B \vdash_{\mathbf{K4}} MK\varphi_i$. If for some j, $\zeta_j \notin T$, then by the induction hypothesis, $\neg K\zeta_j$ is derivable from B. Applying necessitation twice we get $KK\neg K\zeta_j \wedge M\neg\zeta_j$, hence by (i) of the Proposition 7.1, $M(K\neg K\zeta_j \wedge \neg\zeta_j)$, hence $MK\neg K\zeta_j$, hence $MK\varphi_i$.

Thus, the only remaining case is if for each i, j, $\eta_i \notin T$, $\zeta_j \in T$. But then using the stability of T we get $\psi \in T$, ψ is objective, so $MK\psi \in B$, hence $B \vdash_{\mathbf{K4}} MK\varphi_i$. □

Lemma 4.2. *Assume that $\mathcal{S}$ is any logic between* K *and* S5, *T is consistent and satisfies the equation*

$$T = Cn_{\mathcal{S}}(A \cup \{\neg K\varphi : \varphi \in \mathcal{L}_0 \setminus T\} \cup \{\neg K\neg K\psi : \psi \in \mathcal{L}_0 \cap T\}).$$

Then T is stable.

Proof. Clearly T is closed under propositional logic and the necessitation rule. It suffices to prove that for each $\eta \notin T$, $\neg K\eta \in T$. We prove this assertion by induction on $m(\eta)$.

Assume that for each η with $m(\eta) < n$, $\eta \notin T$ implies $\neg K\eta \in T$. Assume that $\varphi \notin T$, $m(\varphi) = n$. We can assume, without loss of generality, that φ is in conjunctive normal form, $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_m$, that each φ_i has a modal depth not larger than n and is a disjunction of atoms, modal atoms and their negations. T is closed under $\mathcal{S}$, hence for some i, $\varphi_i \notin T$. Let φ_i have the form

$$\psi \vee K\eta_1 \vee \ldots \vee K\eta_k \vee \neg K\zeta_1 \vee \ldots \vee \neg K\zeta_l.$$

Clearly, each disjunct here is not in T. T is closed under the necessitation rule, hence, for each i,

$$\eta_i \notin T.$$

By the induction hypothesis, we have

$$\neg K\eta_i \in T,$$

hence

$$K\neg K\eta_i \in T.$$

We have for each j, $m(\zeta_j) < n$, hence, applying the induction hypothesis, we obtain $\zeta_j \in T$, hence $KK\zeta_j \in T$. Now, we have $M\neg\psi \notin T$. Applying Proposition 7.1(i) $k + l$ times, we obtain

$$M(\neg\psi \wedge \neg K\eta_1 \wedge \ldots \wedge \neg K\eta_k \wedge K\zeta_1 \ldots \wedge K\zeta_l) \in T,$$

which means $\neg L\varphi_i \in T$, hence $\neg L\varphi \in T$, which completes the proof. □

For the next proof we need the basics of Kripke models which we assume to be known (see, e.g. [HC84, McD82]). A Kripke model $\mathcal{M}$ is a triple $\langle M, R, V\rangle$, where M is a set of worlds, R is an accessibility relation, and $\{V_\alpha\}_{\alpha \in M}$ is a family of propositional valuations. If R is a universal accessibility relation (that is, for all $\alpha, \beta \in M$, $\alpha R\beta$), then we will write $\langle M, V\rangle$ instead of $\langle M, R, V\rangle$. By $\langle \mathcal{M}, \alpha\rangle \models \psi$ we denote the fact that a formula ψ is true in a world α of $\mathcal{M}$. We write $\mathcal{M} \models \psi$ to denote that for each $\alpha \in M$, $\langle \mathcal{M}, \alpha\rangle \models \psi$.

Proposition 4.4. *For each n, there is a theory A such that A has a* K*-expansion (hence an* T*-expansion) which is not an n-bounded* T*-expansion of A (hence not an n-bounded* K*-expansion of A).*

Proof. By $K^n\varphi$ we denote the formula obtained by writing n L-s in front of φ.

Consider the theory $A = \{MK^{n+1}p \supset p\}$. Clearly $T = E(p)$ is the only K-expansion of A and the only T-expansion of A. We prove that

$$p \notin Cn_T(A \cup \{\neg K\phi : m(\varphi) \leq n,\ \varphi \notin T\}). \quad (9)$$

By the soundness theorem for the modal logic T, it is sufficient to construct a Kripke model with a reflexive accessibility relation (a T-model), in which A and all formulas $\neg K\psi$ with $m(\psi) \leq n, \psi \notin T$ are true, but p is false in some world of the model.

Since T is stable, T is an S5-set, i.e. the set of all formulas true in a Kripke model with the universal accessibility relation, say $\mathcal{M} = \langle M, V\rangle$, where M is the set of worlds and $V = \{V_\alpha\}_{\alpha\in M}$ is the family of valuations. Each world in M is accessible from anywhere in M. Consider the following Kripke model $\mathcal{N} = \langle N, R, U\rangle$. As a set of worlds, N, we put the set consisting of $n+2$ disjoint copies of M plus some distinguished world ω. More formally, we put $N = \{\omega\} \cup \{\langle n, \beta\rangle : 0 \leq n \leq n+1,\ \beta \in M\}$. Put $U_{\langle n,\beta\rangle}(q) = V_\beta(q)$ for each propositional variable q, and put $U_\omega(q) = 0$ for each propositional variable q (including $q = p$). Define the accessibility relation R as follows. Put $\omega R\langle n, \beta\rangle$ iff $n = 1$. Put $\langle m, \beta\rangle R\langle k, \gamma\rangle$ iff $k = 0$ or $k = m$ or $(1 \leq m$ and $k = m+1)$. Put $\langle m, \beta\rangle R\omega$ iff $m = n+1$. Put $\omega R\omega$. Clearly, R is reflexive, so the constructed model is a T-model.

We can describe model $\mathcal{N}$ informally as follows. We have $n+2$ clusters of worlds numbered by numbers $0, \ldots, n+1$. Each of them is a copy of the model $\mathcal{M}$ in which T holds. We have also a distinguished world ω in which p fails. We will call worlds which are in clusters the regular worlds. Worlds in the cluster number 0 we call the final worlds. Worlds in a cluster number $m > 0$ we call worlds of level m. (Note that we do not call the final worlds the "worlds of level 0".) Thus, each world is accessible from any world in the same cluster. Also, from each world of level m, $m \geq 1$, we can access all worlds of the next level (if one exists), as well as "jump" to any final world. From worlds of the maximal level $n+1$ we can either jump to final worlds, or drop to the distinguished world ω. From the final worlds we can access only final worlds. And finally, from the distinguished world ω we can access the level 1 worlds only (it is the only world we cannot access the final worlds from).

First, $MK^{n+1}p \supset p$ is true in all worlds: it is true in all worlds but ω because p is true in those worlds. It is true in ω because $MK^{n+1}p$ is false in ω. (Note that if the final worlds were accessible from ω, then $MK^{n+1}p$ would be true in ω, so the entire implication would be false.)

Next we prove that if $\psi \notin T$ and $m(\psi) \leq n$, then $\neg K\psi$ is true in $\mathcal{N}$. If $\psi \notin T$, then for some world $\alpha \in M$, $\langle \mathcal{M}, \alpha\rangle \models \neg\psi$. Clearly, then $\langle \mathcal{N}, \langle 0, \alpha\rangle\rangle \models \neg\psi$. But $\langle 0, \alpha\rangle$ is a final world, it is accessible from anywhere in N but ω. Thus, for any $\beta \in N$ different from ω, $\langle \mathcal{N}, \beta\rangle \models \neg K\psi$. It remains to prove that $\omega \models \neg K\psi$. Since all 1-worlds are accessible from ω, it suffices to prove that $\langle \mathcal{N}, \langle 1, \alpha\rangle\rangle \models \neg\psi$. The last is an immediate corollary of the following claim.

Claim 7.2 *For any formula η, for any m, $1 \leq m \leq n+1$, if $m(\eta) \leq n+1-m$, then for any m-world $\langle m, \beta\rangle$, $\langle \mathcal{N}, \langle m, \beta\rangle\rangle \models \eta$ if and only if $\langle \mathcal{M}, \beta\rangle \models \eta$.*

Proof. Induction on the complexity of η. The only nontrivial case is in the induction step, when η has the form $K\zeta$. Assume that $\langle \mathcal{N}, \langle m, \beta\rangle\rangle \models K\zeta$. Since all final worlds are accessible from $\langle m, \beta\rangle$, we have for any final world γ, $\langle \mathcal{N}, \gamma\rangle \models \zeta$, hence , obviously, $\langle \mathcal{M}, \beta\rangle \models K\zeta$.

Conversely, assume that $\langle \mathcal{M}, \beta\rangle \models K\zeta$. Then for each γ, $\langle \mathcal{M}, \gamma\rangle \models \zeta$. Hence for each final world γ of N, we have $\langle \mathcal{N}, \gamma\rangle \models \zeta$. We have $m(\zeta) \leq n+1-(m+1) < n+1-m$, hence using the induction hypothesis we have for each $\gamma \in M$, both $\langle \mathcal{N}, \langle m, \gamma\rangle\rangle \models \zeta$ and $\langle \mathcal{N}, \langle m+1, \gamma\rangle\rangle \models \zeta$. Thus for each γ which is accessible from $\langle m, \beta\rangle$, we have $\langle \mathcal{N}, \gamma\rangle \models \zeta$, henceforth $\langle \mathcal{N}, \langle m, \beta\rangle\rangle \models K\zeta$. □

Thus, all formulas $\neg K\psi$ with $\psi \notin T$ and $m(\psi) \leq n$ are true in each world of the model $\mathcal{N}$, as well as $MK^{n+1}p \supset p$ is, but p is false in ω. Thus, (9) has been proved. □

Lemma 5.1. *If T is consistent and stable, then for each $\eta \in T$,*

$$T_0 \cup \{\neg K\psi : \psi \in \mathcal{L}_0 \setminus T\} \vdash_K \eta. \quad (10)$$

Proof. We prove simultaneously by induction on the depth of nesting L in η, that if $\eta \in T$ then (10) holds, and if $\eta \notin T$, then

$$\{\neg K\varphi : \varphi \in \mathcal{L}_0 \setminus T\} \cup T_0 \vdash_K \neg K\eta. \quad (11)$$

Assume that (10) holds for each $\eta \in T$ with $m(\eta) < n$, and (11) holds for each $\eta \notin T$ with $m(\eta) < n$.

Assume that $m(\eta) = n$ and $\eta \notin T$ and prove (11). We may assume without loss of generality that η is in conjunctive normal form, $\eta = \varphi_1 \wedge \ldots \wedge \varphi_m$, that each φ_i has a modal depth not larger then n and is of the form

$$\psi \vee K\eta_1 \vee \ldots \vee K\eta_k \vee \neg K\zeta_1 \vee \ldots \vee \neg K\zeta_l, \quad (12)$$

where ψ is objective and for each i, j, $m(\eta_i) < n, m(\zeta_j) < n$. Clearly, some φ_i is not in T. Assume that such φ_i is of the form (12). Clearly each disjunct in (12) is not in T. T is stable, hence

$$\psi \notin T,\ \eta_i \notin T,\ \zeta_j \in T$$

for each i, j.

Denote $\{K\varphi : \varphi \in \mathcal{L}_0 \setminus T\} \cup T_0$ by B. By the induction hypothesis, the necessitation rule and the definition of B we have

$$B \vdash_{K4} \neg K\psi,\ B \vdash K\neg K\eta_i,\ B \vdash KK\zeta_j$$

for each i, j. Applying (i) of Lemma 7.1 several times and rewriting $\neg K$ as $M\neg$ we get

$$B \vdash_K M(\neg\psi \wedge \neg K\eta_1 \wedge \ldots \wedge \neg K\eta_k \wedge K\zeta_1 \wedge \ldots \wedge K\zeta_l)$$

which may be rewritten as

$$B \vdash_K \neg K(\psi \vee K\eta_1 \vee \ldots \vee K\eta_k \vee \neg K\zeta_1 \vee \ldots \vee \neg K\zeta_l),$$

which is just $B \vdash_K \neg K\varphi_i$ which implies $B \vdash_K \neg K\eta$.

Assume now that $\eta \in T$ and $m(\eta) = n$ and prove (10). Again, we can assume without loss of generality that η is in conjunctive normal form, $\eta = \varphi_1 \wedge \ldots \wedge \varphi_m$, and that each φ_i is of the form (12). Clearly each φ_i is in T. It is sufficient to prove that for each φ_i, $B \vdash_K \varphi_i$.

Assume φ_i has the form (12). If for some j, $\eta_j \in T$, then we have by the induction hypothesis $B \vdash_K \eta_j$, hence using the necessitation rule and K we easily obtain $B \vdash_K \varphi_i$. If for some j, $\zeta_j \notin T$ then by the induction hypothesis, $\neg K\zeta$ is derivable from B, hence $B \vdash_K \varphi_i$.

Thus, the only remaining case is if for each i, j, $\eta_i \notin T$, $\zeta_j \in T$. But then using the stability of T we get $\psi \in T_0$. □

Acknowledgements

Discussions on the subject of the paper with Vladimir Lifschitz, Wiktor Marek and Miroslaw Truszczynski were very useful. Miroslaw Truszczynski, Arcot Rajasekar and James Rogers carefully read earlier versions of the paper and made many valuable comments. The author is grateful to anonymous referees for constructive criticism, which helped in preparing the final version.

Significant part of the work reported here was performed when the author was visiting the University of Kentucky and was supported by NSF and the Commonwealth of Kentucky EPSCoR program under grant RII 8610671.

References

[Che80] B.F. Chellas. *Modal logic, and introduction.* Cambridge University Press, 1980.

[Fey65] R. Feys. *Modal Logics.* Louvain E. Nauwelaerts, Paris, 1965.

[Gel89] M. Gelfond. Autoepistemic logic and formalization of commonsense reasoning. In M. Reinfrank, J. de Kleer, M.L. Ginsberg, and E. Sandewall, editors, *Non-Monotonic Reasoning*, pages 176–186. Springer-Verlag, 1989. Lecture Notes in Artificial Intelligence, 346.

[HC84] G.E. Hughes and M.J. Cresswell. *A companion to modal logic.* Methuen and Co. Ltd., London, 1984.

[Hin62] Jaakko Hintikka. *Knowledge and Belief: an Introduction to the Logic of Two Notions.* Cornell University Press, 1962.

[HM85] J.Y. Halpern and Y. Moses. A guide to modal logics of knowledge and belief. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 480–490. Morgan Kaufmann, 1985.

[HM90] J.Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37:549 - 587, 1990.

[Kam91] M. Kaminski. Embedding a default system into nonmonotonic logic. *Fundamenta Informaticae*, 14:345 - 354, 1991.

[Kon88] K. Konolige. On the relation between default and autoepistemic logic. *Artificial Intelligence*, 35:343–382, 1988.

[Lak92] D. Lakemeyer. On perfect introspection with quantifying-in. In Y. Moses, editor, *Theoretical Aspects of Reasoning about Knowledge. Proceedings of the Fourth Conference (TARK 1992)*, pages 199–213, San Mateo, CA, 1992. Morgan Kaufman.

[Len78] W. Lenzen. *Recent Work in Epistemic Logic*, volume 30 of *Acta Philosophica Fennica.* North-Holland, Amsterdam, 1978.

[Len79] W. Lenzen. Epistemologische Betrachtungen zu [S4,S5]. *Erkenntnis*, 14:33–56, 1979.

[Lif91] V. Lifschitz. Nonmonotonic databases and epistemic queries. In *Proceedings of IJCAI-91*, pages 381–386, Los Altos, CA., 1991. Morgan Kaufmann.

[McD82] D. McDermott. Nonmonotonic logic II: Nonmonotonic modal theories. *Journal of the ACM*, 29:33–57, 1982.

[MD80] D. McDermott and J. Doyle. Nonmonotonic logic I. *Artificial Intelligence*, 13:41–72, 1980.

[Moo84] R.C. Moore. Possible-world semantics autoepistemic logic. In R. Reiter, editor, *Proceedings of the workshop on non-monotonic reasoning*, pages 344–354, 1984. (Reprinted in: M.Ginsberg, editor, *Readings on nonmonotonic reasoning.* pages 137 - 142, 1990, Morgan Kaufmann.).

[Moo85] R.C. Moore. Semantical considerations on non-monotonic logic. *Artificial Intelligence*, 25:75–94, 1985.

[Mor89] P.H. Morris. Autoepistemic stable closure and contradiction resolution. In M. Reinfrank, J. de Kleer, M.L. Ginsberg, and E. Sandewall, editors, *Non-Monotonic Reasoning*, pages 176–186. Springer-Verlag, 1989. Lecture Notes in Artificial Intelligence, 346.

[MS89] Yoram Moses and Yoav Shoham. Belief as defeasible knowledge. In *Proceedings of the IJCAI-89*, pages 1168–1173. Morgan Kauffman, 1989.

[MST91] W. Marek, G.F. Shvarts, and M. Truszczynski. Modal nonmonotonic logics: ranges, characterization, computation. Technical Report 187-91, Department of Computer Science, University of Kentucky, 1991. A revised version is to appear in the *Journal of ACM*.

[Rei78] R. Reiter. On closed world databases. In H. Gallaire and J. Minker, editors, *Logic and data bases*, pages 55–76. Plenum Press, 1978.

[Seg71] K. Segerberg. *An essay in classical modal logic*. Uppsala University, Filosofiska Studier, 13, 1971.

[Shv90] G.F. Shvarts. Autoepistemic modal logics. In R. Parikh, editor, *Proceedings of TARK 1990*, pages 97–109, San Mateo, CA., 1990. Morgan Kaufmann.

[ST92] G. Schwarz and M. Truszczynski. Modal logic S4F and the minimal knowledge paradigm. In Y. Moses, editor, *Proceedings of TARK 1992*, pages 184–198, San Mateo, CA., 1992. Morgan Kaufmann.

[TK90] M. Tiomkin and M. Kaminski. Nonmonotonic default modal logics. In R. Parikh, editor, *Proceedings of TARK 1990*, pages 73–84, San Mateo, CA., 1990. Morgan Kaufmann.

[Tru91a] M. Truszczyński. Modal interpretations of default logic. In *Proceedings of IJCAI-91*, pages 393–398, Los Altos, CA., 1991. Morgan Kaufmann.

[Tru91b] M. Truszczyński. Modal nonmonotonic logic with restricted application of the negation as failure to prove rule. *Fundamenta Informaticae*, 14:355 – 366, 1991.

[Voo91] F. Voorbraak. The logic of objective knowledge and rational belief. In *Logics in AI. European Workshop JELIA '90 Proceedings. Lecture Notes in Artificial Intelligence, v. 478*, pages 499–515. Springer Verlag, 1991.

A Framework for Representing and Characterizing Semantics of Logic Programs

Jürgen Dix
Institute for Logic, Complexity and Deductionsystems,
University of Karlsruhe, POBox 6980
D-W 7500 Karlsruhe, Germany
dix@ira.uka.de

Abstract

In previous work on semantics of logic programs, we have associated to any semantics a nonmonotonic entailment relation " $\sim$ " and considered its abstract properties. We have shown that two important properties, known in the context of nonmonotonic logics, turned out to be useful: *Cumulative* and *Rational Monotony*. Among all known semantics, only the wellfounded semantics WFS satisfies both properties. Our main aim in this paper is to present a set-up which makes it possible to prove statements of the following kind: Any *reasonable* semantics satisfying certain properties is uniquely determined by these.

To this end, we first list some new conditions that should be satisfied by any semantics: *Relevance*, *Modularity*, *Reduction*, *Isomorphy* and *PPE*. A semantics satisfying these (and some more properties) is called *reasonable*. We then present a semantical framework $\mathcal{M}_{NF}$ for defining and comparing semantics of logic programs. This framework is weak enough to represent all of the known semantics of logic programs, including semantics based on Clark's completion. It can be strengthened to ensure that any semantics coincides with the supported semantics of Apt, Blair and Walker when restricted to stratified programs. The system is still weak enough to represent all known extensions of the supported semantics. We argue, that any reasonable semantics fits into our framework.

WFS is the weakest semantics representable in our framework. We claim that a representation theorem holds, showing that there are no cumulative and rational extensions of the stratified semantics other than WFS and a certain extension WFS' of WFS.

1 Introduction

There has been done much work both in the Logic Programming (LP) and in the Nonmonotonic Reasoning (NMR) community for finding the "right" semantics for logic programs containing negation. While one concentrated in LP on variants of SLDNF-resolution and on Clark's completion (which means that a query $? - p$ asked to the program "$p \leftarrow p$" never gets an answer) the intuitions underlying NMR were different: $\neg p$ should follow from "$p \leftarrow p$".

In LP three-valued semantics were introduced to avoid shortcomings of Clark's completion. We showed in [Dix91b] that these are more regular than their two-valued predecessors, because they satisfy *Cumulativity* and *Rationality*.

In the NMR community various other semantics, more suitable for nonmonotonic reasoning tasks, have been proposed: the *supported* semantics M_P^{supp} (see [ABW88]) for *stratified* programs, the *default* "DEFAULT" or *stable* "STABLE" semantics (see [BF89] and [GL88]), and finally the *well-founded* "WFS" semantics (see [vRS88]), the *generalized well-founded* "GWFS" semantics (see [BLM90b]), and various semantics for *disjunctive* programs.

Of all these semantics, the supported semantics of Apt, Blair and Walker plays a fundamental role: all researchers agree, that M_P^{supp} is the only reasonable semantics for stratified programs. Since there are nonstratified programs that also could be assigned a meaning (a unique intended model), the question arises: *What are reasonable extensions of M_P^{supp} to the whole class of programs?*

In previous work ([Dix91a], [Dix91b]) we began to classify semantics of logic programs by considering abstract properties of their induced nonmonotonic entailment relations $\sim$. The overall aim is to get a clear picture of the various existing semantics of logic programs and their mutual relations, by looking at the requirements their $\sim$-relations meet. It turned out that the properties of *Cumulativity* and *Rationality* played

an important role. We already claimed in [Dix91a] that WFS is completely determined by these two properties (our construction of WFS$^+$ and WFS' in section 4 shows that this claim was too hasty). To make such a statement precise, we need a strictly defined semantical framework where any reasonable semantics fits in. Given such a definition, we can formally ask for a description of all cumulative and/or rational semantics and eventually prove a representation theorem of the following kind:

Any semantics, the $\sim$ -relation of which satisfies certain conditions, is uniquely determined by these.

In this paper, we develop such a framework and prove a representation theorem and conjecture a characterization.

Two competing approaches, defined for the class of all normal programs, are the wellfounded semantics WFS and the stable semantics STABLE. Some people argue, that WFS is overly careful and should be extended to also solve the *floating conclusions problem*. $GWFS$ of Baral, Lobo and Minker is such an extension. We show that $GWFS$ has a serious shortcoming (the failure of PPE), but that a more cautious approach yields a cumulative and rational extension of WFS that satisfies PPE: WFS$^+$ and WFS'.

Our method of defining the semantical framework is related to and partially inspired by [KLM90] and [Mak89]. Kraus, Lehmann and Magidor defined various abstract conditions for a relation $\sim$, defined a semantical framework $(\mathcal{U}, \prec)$, and showed various representation theorems. The main difference to our framework is that our $\sim$ -relation is not a relation between arbitrary (sets of) *formulae*, but a relation between (sets of) *atoms* and *literals*. In addition, $\sim$ is always defined with respect to a *fixed* program P: $\sim = \sim_P$.

In Section 2 we introduce some terminology which is needed later on, state some interesting properties of a nonmonotonic relation " $\sim$ " due to Kraus, Lehmann, Magidor and Makinson, and introduce the sceptical entailment SEM_P^{scept} of a semantics SEM.

In Section 3 we motivate the supported semantics M_P^{supp} by showing some shortcomings of the completion approach. We also define the wellfounded semantics WFS and illustrate its properties.

In Section 4 we introduce the stable semantics and the floating conclusions problem (which lead to the definition of GWFS) to motivate the need for extensions of WFS. We also introduce our first condition, PPE, the principle of partial evaluation, and argue that any semantics should satisfy it. We present a program P showing the failure of the PPE for GWFS, and define WFS$^+$ and WFS', two cumulative and rational extension of WFS.

Section 5 summarizes our results from [Dix91a, Dix91b] and shows the relations between the semantics considered so far. These results are important, since they show which kind of representation theorems are possible.

In Section 6 we consider the question "What are reasonble properties that should be satisfied by any semantics?". We introduce and discuss some new conditions and argue, that any reasonable semantics should satisfy them.

In Section 7 we present our semantical framework and illustrate that any of the known semantics fits into it. We also state our main results: characterizations of various semantics by means of simple properties of the partial order $\prec$ used in our framework.

Section 8 indicates that our conditions are also interesting for disjunctive programs and may be easily adapted. An extension of our semantical framework, however, still needs to be worked out.

We end in Section 9 with some additional remarks and a short outlook.

2 Notation and Terminology

A general *normal* logic program consists of rules that allow arbitrary *positive* clauses to appear in their heads:

$$A \leftarrow B_1, \ldots B_m, \neg C_1, \ldots \neg C_l \quad \text{where } n \geq 1 \ .$$

If $l = 0$, the program is *positive* (or *definite*). When we use the notation $P \cup U$, U stands for an (infinite) set of atoms *positive facts* we want to distinguish from the facts in P. That is, P contains rules and positive facts, but U only contains atoms. P_{inst} stands for the (infinite) sets of fully instantiated rules and facts (*ground* clauses). A program is, in addition, called *propositional* or *Datalog*, if it does not contain any function symbols, but only propositional variables and their negations. It is convenient to assume that all programs are fully instantiated, i.e., we consider only propositional programs in this paper. This can be done without loss of generality: all our results generalize to the class of arbitrary programs.

We denote the Herbrand-base with respect to a program P by $B_{\mathcal{L}_P}$, or simply B_P: the underlying language $\mathcal{L}_P$ is given by the symbols in P. $Th(\Phi)$ denotes the classical closure of the set of formulae Φ and Fml denotes the set of all formulae. Let MIN-MOD(T) denote the class of all *two-valued* minimal Herbrand-models of T.

We also need some notions from 3-valued logic. We use truth values t "true", f "false", u "undefined", and the Kleene connectives $\vee, \wedge, \neg$ and $\leftarrow$. $\leftarrow$ is the *weak* implication, where "$u \leftarrow u$" is considered to be *true*. Additionally, we can use two different orderings

of the truth values: the lattice $\mathbf{3}_t$ defined by $f \leq_t u \leq_t t$ (truth-ordering) and the semi-lattice $\mathbf{3}_k$ defined by $u \leq_k t$, $u \leq_k f$ (knowledge-ordering).

Given two 3-valued models $\mathcal{A}$ and $\mathcal{A}'$, we denote by $\mathcal{A} \sqcap_k \mathcal{A}'$ their $\sqcap_k$-intersection:

$$\mathcal{A} \sqcap_k \mathcal{A}' = \{l : l \text{ a literal with } l \in \mathcal{A} \text{ and } l \in \mathcal{A}'\}.$$

We will consider 3-valued models as sets of literals and vice versa. A set $\{a, \neg b, c, \neg d\}$ will therefore also be denoted by $< \{a, c\}; \{b, d\} >$ and can thus be seen as a three-valued structure: the first part describes the atoms that are considered true, the second part describes atoms that are considered to be false. $< T; F > \leq_k < T'; F' >$ therefore means, that $T \subseteq T'$ and $F \subseteq F'$.

When we say that a semantics SEM *extends* another semantics SEM', we have to distinguish between two different notions:

- $\text{SEM}_1 \leq_k \text{SEM}_2$: this means that SEM_2 classifies more atoms as true or false than SEM_1, or
- SEM_2 is defined for a class of programs that strictly includes the class of programs for which SEM_1 is defined and for all programs of this smaller class, the two semantics coincide.

The first notion also makes perfectly sense for semantics defined for the same class of programs.

2.1 Kraus, Lehmann and Magidor' s Rules

The following structural properties for an entailment relation " $\vdash$ " *between single formulae* were considered by Kraus, Lehmann and Magidor:[1]

$$\begin{array}{lllll} \textit{C.M.} & \alpha \vdash \beta & , \quad \alpha \vdash \gamma & \Longrightarrow & \alpha \wedge \beta \vdash \gamma. \\ \textit{Cut} & \alpha \vdash \beta & , \quad \alpha \wedge \beta \vdash \gamma & \Longrightarrow & \alpha \vdash \gamma. \\ \textit{Rat.} & \alpha \not\vdash \neg\beta & , \quad \alpha \vdash \gamma & \Longrightarrow & \alpha \wedge \beta \vdash \gamma. \end{array}$$

The property *Cut* is uncontroversial: it is satisfied by any known nonmonotonic system. *Cautious Monotony* (C. M.) combined with *Cut* is the following condition, called *Cumulativity*

$$\text{If } \alpha \vdash \beta \text{ then: } \alpha \vdash \gamma \text{ iff } (\alpha \wedge \beta) \vdash \gamma .$$

It states that, whenever a fomula β has been derived from α, β can be used as a "lemma" and does not affect the set of formulae derivable from α alone. If this condition is not valid, intermediate lemmas are of no use: this indicates that reasoning in non-cumulative systems may be more expensive.

Obviously, *Rationality* implies *Cautious Monotony*. Since *Cut* is always satisfied, we will in the sequel use the terms "cautious" and "cumulative" interchangeably.

The next section states the precise definitions we are using later on. The greatest difference to the formulations in this section is that we do not have " $\vdash$ " as a relation on the *whole* set of formulae.

2.2 Sceptical Semantics $SEM_P^{scept}(U)$

We now introduce a precise terminology for our abstract notions of *Cumulativity* and *Rationality* in the context of general disjunctive programs and give some general remarks.

Model-theoretically, all the semantics we will be presenting are defined as subsets of $\text{MOD}_3(P \cup U)$ (the set of all 3-valued models of $P \cup U$). More precisely[2]

- $\text{SEM}_P(U) \subseteq \text{MOD}_3(P \cup U)$.

For all semantics it is possible to define a *sceptical* notion $\text{SEM}_P^{scept}(U)$ of entailment for arbitrary literals L:

$$\bigcap_{\mathcal{M} \in SEM_P(U)} \{L : L \text{ is a literal with: } \mathcal{M} \models L\}$$

We also write SEM(P) instead of $\text{SEM}_P^{scept}(\emptyset)$. Thus, SEM(P) can be seen as a three-valued model. We will also use the notation $SEM(P)(x) = t$ (resp. $= f$, resp. $= u$) to mean that $x \in SEM(P)$ (resp. $\neg x \in SEM(P)$, resp. neither x nor $\neg x$ are contained in $SEM(P)$).

Compared with the " $\vdash$ " framework of Kraus, Lehmann and Magidor, each of our semantics induces such a " $\vdash$ "-relation between sets of *atoms* (positive literals) on the left hand side, and sets of arbitrary *literals* on the right hand side:

- $a_1 \wedge \ldots \wedge a_n \vdash l_1 \wedge \ldots \wedge l_m$ **:iff** $\{l_1, \ldots l_m\} \subseteq SEM_P^{scept}(\{a_1, \ldots a_n\})$

Rationality is in any *sceptical* semantics a stronger form of *Cautious Monotony* because "$\alpha \vdash \beta$" implies "not $\alpha \vdash \neg\beta$".

3 $comp(P)$, M_P^{supp} and WFS

comp) In Logic Programming the first semantics to derive *negative* literals from a definite program was defined by Clark: $comp(P)$ (see [Llo87]). $comp(P)$ is a first-order theory but it is already inconsistent for simple programs containing negation[3]. In addition,

[1] " $\vdash$ " can be extended to a relation between *finite* sets of formulae using $\wedge$.

[2] For most semantics, we consider three-valued Herbrand-models of the underlying language $\mathcal{L}_{P \cup U}$.

[3] E.g. "$p \leftarrow \neg p$": In [Prz88a], various problems with *comp* concerning the treatment of loops were discussed.

even for a consistent $comp(P)$, cumulativity does not hold:

Example 3.1 (COMP not cautious)

$$P_{comp}: \begin{array}{lcl} a & \leftarrow & \neg p \\ a & \leftarrow & b \\ p & \leftarrow & a \\ p & \leftarrow & b \\ b & \leftarrow & b \end{array} \qquad comp(P): \begin{array}{lcl} a & \leftrightarrow & \neg p \vee b \\ p & \leftrightarrow & a \vee b \\ b & \leftrightarrow & b \end{array}$$

$comp(P_{comp})$=$Th(\{a, b, p\})$ implies a, b and p. But $comp(P_{comp} \cup \{p\})$ is the conjunction of $a \leftrightarrow (\neg p \vee b)$ and $p \leftrightarrow (a \vee b \vee true)$, so it is exactly $Th(\{p, a \leftrightarrow b\})$, from which a (nor b) no longer follows.

Fitting considered $comp_3(P)$, a three-valued formulation of the completion, to solve this inconsistency problem and showed that there always exists a unique $\leq_k$-least, minimal (three-valued) Herbrand model of $comp_3(P)$. This model is the least fixpoint of a $\leq_k$-monotone operator Φ:

$$\Phi_P : \mathbf{3}_k{}^{B_P} \longmapsto \mathbf{3}_k{}^{B_P};\ \mathcal{I} \longmapsto \Phi_P(\mathcal{I})$$

$$\Phi_P(\mathcal{I})(A) = \begin{cases} \mathbf{t}, \text{ if there is a clause} \\ \quad A \leftarrow L_1, \ldots, L_n \text{ in } P_{inst} \text{ with:} \\ \quad - \ \forall i \leq n \text{ we have:} \\ \quad\quad * \ \mathcal{I}(L_i) = \mathbf{t}. \\ \mathbf{f}, \text{ if for all clauses } A \leftarrow L_1, \ldots, L_n \\ \quad \in P_{inst} \text{ we have:} \\ \quad - \ \exists i \leq n \text{ with:} \\ \quad\quad * \ \mathcal{I}(L_i) = \mathbf{f}. \\ \mathbf{u} \text{ otherwise} \end{cases}$$

But there is still a problem with this three-valued completion: From the standpoint of nonmonotonic reasoning, programs containing positive loops, such as

$$\begin{array}{lcl} p & \leftarrow & p \\ a & \leftarrow & \neg p \end{array}$$

were not handled correctly: while $comp(P)$ does not derive $\neg p$ nor a, any reasonable semantics should do this, since p can never be derived. This intuition can be extended to programs of the form

$$P_{strat}: \begin{array}{lcl} p & \leftarrow & \neg q \\ q & \leftarrow & \neg b \\ b & \leftarrow & b \end{array}$$

Note that p depends negatively on q, q depends negatively on b and b does not depend negatively on another atom. It seems reasonable to first minimize b, then q and finally p. This yields the model $\{\neg b, q, \neg p\}$. The important notion to formalize this intuitive reasoning is the dependency-graph:

Definition 3.2 (Dependency-Graph $\mathcal{G}_P$)
For a normal logic program P, the dependency graph *$\mathcal{G}_P$ is an infinite directed graph whose set of vertices is the Herbrand-base B_P. There is a* positive *(respectively* negative*) edge from a to a' :iff there is a clause in P with a in its head and a' occurring positively (respectively negatively) in its body.*

We say that

- *a* depends on *a' if there is a path in $\mathcal{G}_P$ from a to a' (by definition, a depends on itself),*
- *a depends negatively on a' if there is a path in $\mathcal{G}_P$ from a to a' containing at least one negative edge.*

A program is called *stratified*, if there are no atoms depending negatively on themselves: the supported semantics M_P^{supp} of Apt, Blair, and Walker (which is cumulative and rational, see [Dix91a]) is defined for this class of programs. Their definition is analogous to our construction of $\{\neg b, q, \neg p\}$, which is the supported model of P_{strat}.

The wellfounded semantics WFS extends the supported semantics to the class of all logic programs. We need an extension of Φ_P:

$$\Phi_P^{\mathcal{J}} : \mathbf{3}_t{}^{B_P} \longrightarrow \mathbf{3}_t{}^{B_P};\ \mathcal{I} \longmapsto \Phi_P^{\mathcal{J}}(\mathcal{I})$$

$$\Phi_P^{\mathcal{J}}(\mathcal{I})(A) = \begin{cases} \mathbf{t}, \text{ if there is a clause} \\ \quad A \leftarrow L_1, \ldots, L_n \text{ in } P_{inst} \text{ with:} \\ \quad - \ \forall i \leq n \text{ we have:} \\ \quad\quad * \ \mathcal{J}(L_i) = \mathbf{t}, \\ \quad\quad\quad \underline{\text{or}} \\ \quad\quad * \ L_i \text{ is positive, } \mathcal{J}(L_i) \neq \mathbf{f} \\ \quad\quad\quad \text{and } \mathcal{I}(L_i) = \mathbf{t}. \\ \mathbf{f}, \text{ if for all clauses } A \leftarrow L_1, \ldots, L_n \\ \quad \in P_{inst} \text{ we have:} \\ \quad - \ \exists i \leq n \text{ with:} \\ \quad\quad * \ \mathcal{J}(L_i) = \mathbf{f} \\ \quad\quad\quad \underline{\text{or}} \\ \quad\quad * \ L_i \text{ is positive, } \mathcal{J}(L_i) \neq \mathbf{t} \\ \quad\quad\quad \text{and } \mathcal{I}(L_i) = \mathbf{f}. \\ \mathbf{u} \text{ otherwise} \end{cases}$$

$\mathcal{J}$ represents the *safe* knowledge: True($\mathcal{J}$) and False($\mathcal{J}$) never decrease. $\mathcal{I}$ stands for the current knowledge; applying $\Phi_P^{\mathcal{J}}$ to $\mathcal{I}$ and iterating this further gives us new information (False($\mathcal{I}$) decreases). Positive and negative information is handled symmetrically[4].

Przymusinski proved that for all three-valued interpretations $\mathcal{J}$, the operator $\Phi_P^{\mathcal{J}}$ is *$\leq_t$-monotone* on $\mathbf{3}_t{}^{B_P}$ and has a *unique $\leq_t$-least fixpoint*:

$$\text{lfp}(\Phi_P^{\mathcal{J}}) = \Phi_P^{\mathcal{J}}\uparrow^{\omega}, \text{where } \Phi_P^{\mathcal{J}}\uparrow^0 = < \emptyset, B_P > .$$

Note, that $\Phi_P^{\mathcal{J}}$ is also $\leq_k$-monotone.

[4] As before, negative literals are only evaluated by $\mathcal{J}$!

He then defined the $\leq_k$-monotone operator Ω_P

$$\Omega_P; \mathbf{3}_k{}^{B_P} \longmapsto \mathbf{3}_k{}^{B_P} : \mathcal{J} \longmapsto \Phi_P^{\mathcal{J}}\uparrow^\omega .$$

Ω_P is also $\leq_k$-monotone, thus it has a unique $\leq_k$-least fixpoint, which is exactly the well-founded model from *van Gelder, Ross and Schlipf*:

$$WFS(P) = lfp(\Omega_P).$$

This least fixpoint is a consistent three-valued model, because it is the iteration of a consistency preserving operator.

4 STABLE and Extensions of WFS

To define the stable semantics (and also to formulate our principles of *Modularity* and *Reduction* in the next sections), we need the notion of

reducing a program using a set of literals M.

The following definition captures this idea

Definition 4.1 (P reduced by M)
"P reduced by M" is the program

$$P^M := \{rule^M : rule \in P\},$$

where $(A \leftarrow B_1, ..., B_n, \neg C_1, ...\neg C_m)^M$ is defined by

$$\begin{cases} true & \text{if } \exists j : C_j \in M \text{ or } \neg B_j \in M \\ rule' & \text{otherwise} \end{cases}$$

Here, $rule'$ stands for the clause "$A \leftarrow B'_1, ..., B'_{n'}, \neg C'_1, ...\neg C'_{m'}$", where the set $\{B'_i : i \in I'\}$ (resp. $\{\neg C'_i : i \in I'\}$) is just an enumeration of the set $\{B_i : B_i \in I\} \setminus M$ (resp. $\{\neg C_i : \neg C_i \in I\} \setminus M$).

This notion of reduction closely resembles the Gelfond-Lifschitz transform to define stable models. The difference is, that Gelfond-Lifschitz reduce with a set N of *atoms* and implicitly assume that all atoms not in N are false. We are more cautious and reduce only those literals, that are explicitly contained in M. In addition, our reduced program still contains negation: the Gelfond-Lifschitz transform is a positive program. Using our reduction, the stable models of a program P can be defined as follows:

Definition 4.2 (STABLE)
$N \subseteq B_P$ is a stable *model of P, if the least Herbrand model $M_{P^{N \cup \neg N}}$ of "P reduced by $N \cup \neg N$" equals N.*

STABLE(P) is therefore the set of all literals true in all stable models. A serious shortcoming of STABLE (besides the failure of cumulativity: see [Dix91a]) is the failure of *Relevance* as discussed in the next section. This is due to the fact that stable models need not always exist: programs containing negative loops "$p \leftarrow \neg p$" do not have stable models. On the other hand, some people argue that STABLE behaves more regularly on examples where *floating conclusions* play a role. This can be motivated by the following program which shows that any sceptical semantics based on the intersection of two-valued models (like STABLE) inherits disjunctions:

Example 4.3 (Floating Conclusions)

$$\begin{array}{lrcl} P_1: & a & \leftarrow & \neg b \\ & b & \leftarrow & \neg a \\ & p & \leftarrow & a \\ & p & \leftarrow & b \end{array}$$

Although neither a, nor b can be "derived" in any semantics based on two-valued models (as STABLE for example), the disjunction $a \vee b$, thus also p, is true: $STABLE(P_1) = <\{p\}; \emptyset>$ $WFS(P_1)$, however, is empty: $WFS(P_1) = <\emptyset; \emptyset>$.

The simplest way to augment WFS for this task is by using minimal model reasoning. There are, however, various possibilities. The simplest one is first evaluating WFS and then adding all literals, that are true in all minmal models extending WFS(P): this defines EWFS, a precursor of WFS$^+$ defined in [Dix91b]. Now, this procedure may add new literals in such a way, that a further application of WFS still yields new literals: *Cut* is not satisfied. In order to get some *"closure"* of this process, Baral, Lobo and Minker defined GWFS (see [BLM90b]). While GWFS satisfies *Cut* but not *Cautious Monotony*, EWFS is cautious but fails to satisfy *Cut*. GWFS has another strange behaviour: it does not satisfy the *principle of partial evaluation*. We now give the precise notions.

The exact realization of the intuitive idea of extending WFS by adding all literals true in all minimal models extending WFS is by using the operator

$$\Omega_P; \mathbf{3}_k{}^{B_P} \longrightarrow \mathbf{3}_k{}^{B_P} : \mathcal{J} \longmapsto \Phi_P^{\mathcal{J}}\uparrow^\omega,$$

which is based on the definition of

$$\Phi_P^{\mathcal{J}} : \mathbf{3}_t{}^{B_P} \longrightarrow \mathbf{3}_t{}^{B_P};\ \mathcal{I} \longmapsto \Phi_P^{\mathcal{J}}(\mathcal{I})$$

as defined in the last section.

Note, that WFS(P):= lfp(Ω_P). GWFS is then defined as follows:

Definition 4.4
For a three-valued interpretation $\mathcal{J}$, let $\mathcal{J}$–MIN-MOD(P) denote the class of all minimal two-valued Herbrand models of P that are consistent with $\mathcal{J}$. Furthermore, let

- $\mathbf{T}(\mathcal{J}) :=$ *True($\mathcal{J}$–MIN-MOD(P)) and*
  $\mathbf{F}(\mathcal{J}) :=$ *False($\mathcal{J}$–MIN-MOD(P)).*

Here, for a set $\mathcal{S}$ of Herbrand models, True($\mathcal{S}$) (resp. False($\mathcal{S}$)) stand for the set of all ground atoms A, which are true (resp. the negations of which are true) in all models in $\mathcal{S}$.

We define

$$EWFS(P) := WFS(P) + <\mathbf{T}(WFS(P)), \mathbf{F}(WFS(P))>.$$

$$GWFS(P) := lfp(\Omega_P^G),$$

where the operator Ω_P^G *is defined as follows:*

$$\Omega_P^G; \mathbf{3}_k{}^{B_P} \longmapsto \mathbf{3}_k{}^{B_P} : \mathcal{J} \longmapsto \Phi_P^{\mathcal{J}} \uparrow^\omega + <\mathbf{T}(\mathcal{J}), \mathbf{F}(\mathcal{J})>.$$

EWFS is stronger than WFS (WFS $\leq_k$ EWFS) and it is also cautious (see [Dix91a]). That it does not satisfy *Cut* can be seen with the following counterexample:

$$\begin{array}{lrcl} P: & a & \leftarrow & \neg a \\ & \beta & \leftarrow & \neg x, a \\ & y & \leftarrow & \neg \beta \\ & z & \leftarrow & \neg y \end{array}$$

We have WFS(P)=$\{\neg x\}$ and therefore EWFS(P)=$\{a, \beta, \neg x\}$ but WFS($P \cup \{\beta\}$)=$\{\neg y, z, \beta\}$ and therefore EWFS($P \cup \{\beta\}$) = $\{a, \beta, \neg x, \neg y, z\}$. Thus, by adding a derivable atom ("β"), we are able to derive more atoms than without: *Cut* is not satisfied.

The failure of *Cautious Monotony* for GWFS can be illustrated with the following

Example 4.5 (GWFS is not cautious)

$$\begin{array}{lrcl} P_2: & p & \leftarrow & \neg b \\ & b & \leftarrow & c \\ & c & \leftarrow & p, \neg a \\ & a & \leftarrow & \neg b \end{array}$$

GWFS(P_2) entails $\neg c$*, because MIN-MOD(P_2) consists of* $\{\ \{p, a\},\ \{b\}\ \}$ *and thus also* $\neg b$*,* p *and* a*. But MIN-MOD($P_2 \cup \{p\}$)=$\{\ \{p, a\},\ \{p, c, b\}\ \}$, thus GWFS($P_2 \cup \{p\}$) does not entail* $\neg c$*.*

Note that intuitively, P_2 can be seen as an *extension-by-definition* of the program P_{2c}, resulting from P_2 by amalgamating the two clauses $b \leftarrow c$ and $c \leftarrow p, \neg a$ into $b \leftarrow p, \neg a$, but GWFS(P_2') neither implies p nor a. We consider this to be a serious shortcoming.

$$\begin{array}{lrcl} P_2: & p & \leftarrow & \neg b \\ & b & \leftarrow & c \\ & c & \leftarrow & p, \neg a \\ & a & \leftarrow & \neg b \end{array} \qquad \begin{array}{lrcl} P_{2c}: & p & \leftarrow & \neg b \\ & b & \leftarrow & p, \neg a \\ & a & \leftarrow & \neg b \end{array}$$

Note that c occurs only positively in P_2 and all occurences have been replaced by the rule defining it, but yet the semantics of P_2 and P_{2c} differ: P_2 derives p and a while P_{2c} does not.

Our principle of partial evaluation, PPE, states that this should not happen: any semantics should assign the same meaning to a program P and a partial evaluation of it.

The formal definition is as follows:

Definition 4.6 (PPE)
Let P be an instantiated program and let the atom c occur only positively in P. Let $c \leftarrow rhs_1, \ldots c \leftarrow rhs_n$ be all the rules of P with c in their heads. Assume further that $rhs_1, \ldots rhs_n$ do not contain c.

We denote by P_c the program obtained from P by deleting all rules with c in their heads and replacing each rule "$head \leftarrow c, body$" containing c by the rules:

$$\begin{array}{lcl} head & \leftarrow & rhs_1, body \\ \vdots & & \\ head & \leftarrow & rhs_n, body \end{array}$$

The principle of partial evaluation PPE states, that $SEM(P_c) = SEM(P) \setminus \{c, \neg c\}$.

PPE can significantly decrease the complexity of computing a semantics, since it allows us to *reduce* the program by making the underlying Herbrand base smaller: in general, less assignments of elements of B_P have to be taken into account. We argue that PPE should be satisfied by any reasonable semantics.

EWFS satisfies PPE, but *Cut* and *Rationality* do not hold. On the other hand, GWFS adds too much literals. There is, however, also the possibility to add only *positive* literals true in all minimal models[5] (after the computation of WFS) and iterating this further until no new literals can be derived. This yields a semantics WFS$^+$ that extends WFS but still is cumulative and rational.

Definition 4.7 (WFS$^+$)
Let P be a normal program, $M_0 := \emptyset$ and

$$M_{i+1} := \mathbf{T}(WFS(P \cup M_i)).$$

The sequence M_i is clearly increasing and there is an γ such that $M_\gamma = M_{\gamma+1}$. We define

$$WFS^+(P) := WFS(P \cup M_\gamma)$$

Thus, the idea is to add, in contrast to GWFS, only positive atoms true in all twovalued *minimal* models of P that extend $WFS(P \cup M_i)$.

To prove rationality and cumulativity of WFS$^+$, we need first a technical lemma stating some useful properties of WFS$^+$. Let us consider the following simple property of the wellfounded semantics. WFS(P) is always a threevalued model of P, but not necessarily a twovalued one. It is, however, possible to extend WFS(P) to a twovalued model of P: just declare all atoms a with neither $a \in WFS(P)$ nor $\neg a \in WFS(P)$ to be true. The resulting twovalued model is obviously a model of P, since in the head of the rules only positive atoms occur. We have therefore:

$$P \models x \text{ implies } \neg x \notin WFS(P) \quad (*)$$

[5] this amounts obviously to adding all atoms true in $Th(P)$

Lemma 4.8 (Properties of WFS and WFS$^+$)
a) *In definition 4.7, we have:*

$$WFS(P \cup M_i) \leq_k WFS(P \cup M_{i+1})$$

b) *Let M, M' be sets of atoms with $M \subseteq M'$.*

$$\forall m' \in M' : \neg m' \notin WFS^+(P \cup M))$$
$$\textit{implies}$$
$$\forall x' \in \mathbf{T}(WFS(P \cup M')) :$$
$$\neg x' \notin WFS^+(P \cup \mathbf{T}(WFS(P \cup M)))$$

Proof: **a)** The proof is immediate by rationality of WFS (see [Dix91b]).
b) The assumption implies in particular $WFS(P \cup M) \leq_k WFS(P \cup M')$ by rationality of WFS (and since WFS^+ is an extension of WFS). We have therefore $\mathbf{T}(WFS(P \cup M)) \subseteq \mathbf{T}(WFS(P \cup M'))$.

We now prove that

$$WFS(P \cup \mathbf{T}(WFS(P \cup M))) \leq_k$$
$$\leq_k WFS(P \cup \mathbf{T}(WFS(P \cup M'))).$$

Again, by rationality of WFS, it suffices to prove $\forall x' \in \mathbf{T}(WFS(P \cup M'))$:

$$\neg x' \notin WFS(P \cup \mathbf{T}(WFS(P \cup M))).$$

For atoms x' that are already true in $WFS(P \cup M')$ this is immediate, since we have $WFS(P \cup M) \leq_k WFS(P \cup M')$. Let therefore x' be undefined in $WFS(P \cup M')$. If $\neg x' \in WFS(P \cup \mathbf{T}(WFS(P \cup M)))$ then (recall that, by assumption, $\forall m' \in M' : \neg m' \notin WFS^+(P \cup M)$) we have $\neg x' \in WFS(P \cup M' \cup \mathbf{T}(WFS(P \cup M))) \leq_k WFS(P \cup \mathbf{T}(WFS(P \cup M')))$: this would contradict the assumption $x' \in \mathbf{T}(WFS(P \cup M'))$.

This argument can be iterated to prove

$$WFS(P \cup \mathbf{T}(WFS(P \cup \mathbf{T}(WFS(P \cup M))))) \leq_k$$
$$\leq_k WFS(P \cup \mathbf{T}(WFS(P \cup \mathbf{T}(WFS(P \cup M'))))))$$

and so on. This gives us

$$WFS^+(P \cup M) \leq_k WFS^+(P \cup M')$$

from which the condition to be proved follows.

■

Theorem 4.9 (Rationality of WFS$^+$)
$WFS^+(P)$ is a rational extension of WFS that satisfies Cut, i.e. a cumulative and rational extension of WFS.

Proof: We first consider Rationality. Without loss of generality, it suffices to prove "not $\vdash \neg\beta$ and $\vdash \gamma$ imply $\beta \vdash \gamma$". We denote by M_i the sets with respect to P and by M_i^β the sets with respect to $P \cup \{\beta\}$. We show by induction on i:

$M_i \subseteq M_i^\beta$ and $\forall m \in M_i^\beta \cup \{\beta\} : \neg m \notin WFS^+(P \cup M_i)$

For $i = 0$, we have obviously $M_0 = \emptyset = M_0^\beta$ and the whole statement is trivially satisfied.

For $i + 1$, we have $M_{i+1} := \mathbf{T}(WFS(P \cup M_i))$ and $M_{i+1}^\beta := \mathbf{T}(WFS(P \cup M_i^\beta))$. By induction hypothesis, $WFS(P \cup M_i) \leq_k WFS(P \cup \{\beta\} \cup M_i^\beta)$, therefore $M_{i+1} \subseteq M_{i+1}^\beta$. Again by induction hypothesis, we can apply lemma 4.8 b) and get the desired result.

By rationality of WFS, our condition implies

$$WFS(P \cup M_i) \leq_k WFS(P \cup \{\beta\} \cup M_i^\beta)$$

from which $WFS^+(P) \leq_k WFS^+(P \cup \{\beta\})$ follows.

Let us now prove Cut. Let $\vdash \beta$, i.e. there is a γ such that $\beta \in M_\gamma$. We show by induction on i

$$M_i^\beta \subseteq M_{\gamma+i}, \forall m \in M_{\gamma+i} : \neg m \notin WFS^+(P \cup \{\beta\} \cup M_i^\beta)$$

For $i = 0$ we have to prove $\forall m \in M_\gamma : \neg m \notin WFS^+(P \cup \{\beta\})$. Let $m \in M_\gamma$, in particular $m \in WFS^+(P)$. By rationality of WFS^+ and since $\neg\beta \notin WFS^+(P)$, we have $m \in WFS^+(P \cup \{\beta\})$ and therefore $\neg m \notin WFS^+(P \cup \{\beta\})$.

For $i + 1$ by induction hypothesis $WFS(P \cup M_i^\beta) \leq_k WFS(P \cup M_{\gamma+i})$ so that $M_{i+1}^\beta \subseteq M_{\gamma+i+1}$. The induction hypothesis also allows us to apply lemma 4.8 b) to get the desired result.

By rationality of WFS, our condition implies

$$WFS(P \cup \{\beta\} \cup M_i^\beta) \leq_k WFS(P \cup M_{\gamma+i})$$

which implies $WFS^+(P \cup \{\beta\}) \leq WFS^+(P)$.

■

WFS^+ derives a from programs containing "$a \leftarrow \neg a$". A more sceptical variant where such literals are not derivable (they are assigned u) can be obtained by considering not all minimal models extending WFS, but only models $\mathcal{A}'$ that satisfy certain conditions C1 - C4 (we discuss them in the next section) and are $\leq_k$-maximal with respect to

$$WFS(P \cup M_i)) \leq_k \mathcal{A}' \leq_k \mathcal{A} \in \mathbf{T}(WFS(P \cup M_i)).$$

We denote the respective semantics by WFS'. It can be shown along the same lines, that WFS' is a cumulative and rational extension of WFS.

5 The Relations Between the Semantics

The program P_1 in Section 4 shows that STABLE $\not\leq_k$ WFS. We motivate in the next Section, that it is much more regular to define STABLE(P) to be equal to WFS(P) in case there are no stable models. Due to this modification we also have

$$WFS \leq_k STABLE.$$

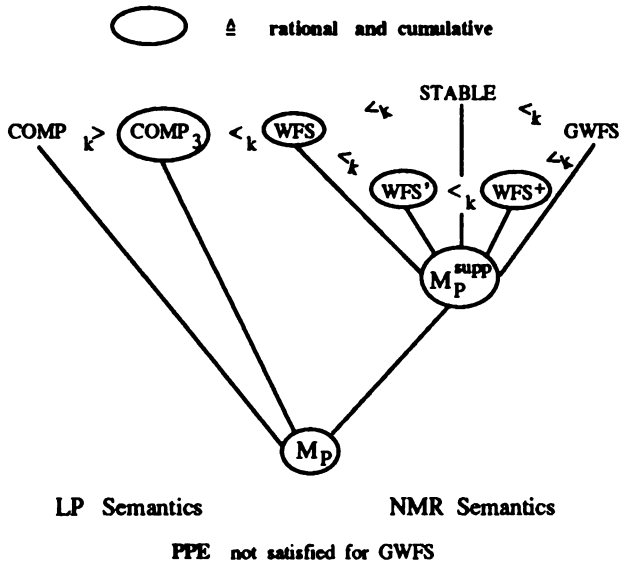


Figure 1: Semantics for Normal Logic Programs

As to normal programs, we can summarize our results from [Dix91b, Dix91a] (and the properties of WFS$^+$ obtained in this paper) in Figure 1.

It is obvious that WFS and COMP are incomparable with respect to $\leq_k$, since already simple loops are differently handled. For the program P consisting of $p \leftarrow p$, we have

$$comp(P) = Th(\{p \leftarrow p\}) \not\leq_k \{\neg p\} = WFS(P)$$

while for P_{comp} (see Example 3.1) we get

$$WFS(P_{comp}) = \{\neg b\} \not\leq_k \{a, b, p\} = comp(P_{comp}).$$

It is also obvious that $\text{COMP}_3 \leq_k WFS$ and $\text{COMP}_3 \leq_k COMP$ since WFS(P) is a three-valued model of P and COMP_3 is the least such one: the definition of $\leq_k$ is important here.

All these relations are illustrated in Figure 1 together with some more relations concerning WFS$^+$ and WFS'.

6 What Is a Reasonable Semantics?

All the known semantics (see Figure 1) are defined by declaring some of the (three-valued) models of P as the intended models. It turns out that the supported model, the stable models (which are two-valued) and also the wellfounded model satisfy the following conditions (this was noted by L. Pereira and explicitly stated in [PAA92]): (let $\mathcal{A}$ be such a model)

C1: If $a \leftarrow rhs \in P$ and $\mathcal{A} \models rhs$, then $\mathcal{A} \models a$.

C2: If $\mathcal{A} \models a$ then there is $a \leftarrow rhs \in P$ with $\mathcal{A} \models rhs$.

C3: If all rule bodies for a are false in $\mathcal{A}$ then $\mathcal{A} \models \neg a$.

C4: If $\mathcal{A} \models \neg a$ then all rule bodies for a are false in $\mathcal{A}$.

Note that we are always interested in the associated sceptical semantics SEM^{scept} which is given by the $\sqcap_k$-intersection of all intended models. We want to list reasonable properties of SEM^{scept} in order to describe particular semantics as uniquely determined by these properties. Any sceptical semantics SEM^{scept} such that SEM is based on models satisfying C1 - C4, also satisfies C1 and C3 but not necessarily C2 and C4. We consider C1 and C3 as reasonable properties that should be satisfied. WFS$^+$ (and also GWFS) do not satisfy C3 because $\text{WFS}^+(\{a \leftarrow \neg a\}) = \{a\}$. WFS' however satisfies C3.

From now on, when we use the notion SEM, we mean the sceptical version, given as a set of literals.

We already mentioned in the last section, that PPE is not satisfied for GWFS. There are still other interesting properties that should be satisfied by any reasonable semantics. To state these conditions, we need the notions of the "Dependency-Graph" and of P reduced by M as defined in the previous section. The condition *Relevance*, also uses the two notions:

- $dependencies\_of(X) := \{A : \text{X depends on A}\}$, and
- $rel\_rul(P, X)$ is the set of *relevant rules* of P with respect to X, i.e. the set of rules that contain an $A \in dependencies\_of(X)$ in their head.

Given any semantics SEM and a program P, it is perfectly reasonable that the truthvalue of a literal L, with respect to SEM(P), only depends on the subprogram formed from the *relevant rules* of P with respect to L.[6] This idea is formalized by:

Definition 6.1 (Relevance)
The principle of Relevance states, that for all literals L: $SEM(P)(L) = SEM(rel\_rul(P, L))(L)$.

Note that the set of relevant rules of a program P with respect to a literal L contains all rules, that could ever contribute to L's derivation (or to its nonderivability). In general, L depends on a large set of atoms: $dependencies\_of(L) := \{A : \text{L depends on A}\}$. But rules that do not contain these atoms in their heads, will never contribute to their derivation or non-derivation. Therefore, these rules should not affect the meaning of L in P.

STABLE does not satisfy this principle. This is due to the nonexistence of stable models by adding a clause "$c \leftarrow \neg c$" to a program. Let us consider the program

$$P = \{\, a \leftarrow \neg b \,\}$$

[6] let $dependencies\_of(\neg X) := dependencies\_of(X)$, and $rel\_rul(P, \neg X) := rel\_rul(P, X)$.

and P' obtained from P by adding "$c \leftarrow \neg c$"

$$P' = \{ a \leftarrow \neg b,\ c \leftarrow \neg c \}.$$

While P has the unique stable model $\{a\}$, P' has no stable models and therefore does not derive $\neg b$ (or, depending on how one interprets the empty set, derives anything). We consider this a very undesirable feature of STABLE. This can be repaired be modifying STABLE when no stable models exist and defining STABLE'(P) to be

$$\begin{cases} WFS(P) & \text{if no stable models exist} \\ STABLE(P) & \text{otherwise} \end{cases}$$

The next principle, *Modularity*, has some similarities with PPE. It enables us to compute a semantics by modularizing it into certain "subprograms" (formed of the relevant rules). The semantics of these modules can be computed first and the semantics of the whole program can be determined by reducing this program with literals that were already determined.

Definition 6.2 (Modularity)
Let P be instantiated, $P = P_1 \cup P_2$, $B_{P_1} \cap B_{P_2} = \{A\}$ and $P_2 = rel\_rul(P, A)$.
Then: $SEM(P) = SEM(P_1^{SEM(P_2)} \cup P_2)$.

We showed in [Dix92a] that the stationary approach of Przymusinski does not satisfy Modularity. We do not know of any semantics for normal programs that fails to satisfy Modularity. Note that, in conjunction with the next property, Modularity implies: If $A \in SEM(P_2)$, then $SEM(P) = SEM(P_1 \cup \{A\}) \cup SEM(P_2)$ (under the assumptions made in the last definition).

There is another natural property satisfied for all semantics in the literature. Suppose we add a set N of atoms to a program P. Independently of whether some of the atoms or their negations are derivable from P alone or not, the semantics of $P \cup N$ should be the same as the semantics of "P reduced by N", augmented by N. The formal statement is

Definition 6.3 (Reduction)
Given a set of atoms $N \subseteq B_P$, the principle of Reduction states that $SEM(P \cup N) = SEM(P^N) \cup N$.

A straigthforward generalization would be to allow not only sets N of atoms, but arbitrary sets M of literals: this possibility will be worked out elsewhere.

We still need a condition to ensure that *equivalent* programs are assigned the same semantics. But this notion of equivalence depends on the underlying semantics: we therefore have to model a very weak form of equivalence, true for all possible semantics. Consider two programs P and Q. If Q is just a renaming of the constants in P, then P and Q should obviously be equivalent, i.e. the renaming of SEM(Q) should equal SEM(P).

Formalizing the equivalence of two programs P and Q (SEM(P)=SEM(Q)) by

"$MOD_3(P)$ is isomorphic to $MOD_3(Q)$",

i.e. the associated classes of all three-valued models are isomorphic, is, however, too strong: the two programs P, defined by $p \leftarrow \neg q$, and Q, defined by $q \leftarrow \neg p$, are not equivalent[7], but yet their classes of three-valued models are even identical: $MOD_3(P) = MOD_3(Q)$. We therefore need, in some way, the *dependencies* between the atoms.

But again, the dependency-graph alone is not the right tool, even for stratified programs: $\mathcal{G}_P$ may be isomorphic to $\mathcal{G}_Q$ without P and Q being equivalent. Just consider P consisting of $a \leftarrow b$, $a \leftarrow \neg c$, $b \leftarrow b$ and Q consisting of $a \leftarrow b, \neg c$ and $b \leftarrow b$: SEM(P)=$\{\neg b, \neg c, a\}$ and SEM(Q)=$\{\neg b, \neg c, \neg a\}$. The examples suggest that the pair $(\mathcal{G}_P, Mod_3(P))$ is a better tool, but again, we can construct counterexamples.

The next condition, *Isomorphy*, formalizes the minimal intuition:

Definition 6.4 (Isomorphy)
Let P and Q be isomorphic, i.e. there is an isomorphism I from $B_{\mathcal{L}_P}$ onto $B_{\mathcal{L}_Q}$ s.t. $I(P) = Q$.
The principle of Isomorphy states that $I(SEM(P)) = SEM(Q)$.

Together with the property *Relevance* and our PPE-principle, Isomorphy also applies to more complex programs that need not be isomorphic but consist of isomorphic subprograms and thus can also be proved to be equivalent.

We are now in a position to define what a reasonable semantics is:

Definition 6.5 (Reasonable Semantics)
A reasonable semantics SEM is a mapping

$$\{P : P \text{ a program}\} \longrightarrow 2^{Lit}$$

such that the following conditions are satisfied: Relevance, Reduction, Isomorphy, Modularity, PPE and Cut. In addition SEM(P) should be a three-valued model of P, satisfy C3 and coincide with M_P^{supp} for stratified programs.

Further work is needed to derive other properties that are implied by our conditions. Our idea is that fixing a semantics on a *reasonable* class of programs (for example axiomatically requiring SEM(P_1)=$\emptyset$, SEM($\{a \leftarrow \neg b, b \leftarrow \neg c, c \leftarrow \neg a\}$)=$\emptyset$ and further axioms) already determines these semantics using our conditions. It is, however, possible that still more conditions have to be used.

[7] SEM(P)=$\{p, \neg q\}$ and SEM(Q)=$\{q, \neg p\}$

7 The Semantical Framework

For any program P we define *not_in_head*(P) as the set consisting of all atoms that do not occur in any head of P.

We are now able to present our semantical framework:

Definition 7.1 ($\mathcal{M}_{NF}$)
Let $NF\text{-}Mod(P)$ be the subset of all 3-valued models $\mathcal{A}$ of P that satisfy C1 - C4. Let $\prec_P$ be a partial order on $NF\text{-}Mod(P)$, such that for every $\mathcal{A}$ there is a $\mathcal{B} \prec_P \mathcal{A}$ that is $\prec_P$-minimal. We define

$$\mathcal{M}_{NF} := \{< NF\text{-}Mod(P), \prec_P >: P \text{ a program}\}.$$

Definition 7.2 (SEM induced by $\mathcal{M}_{NF}$)
The semantics SEM induced by $\mathcal{M}_{NF}$ associates with any program P the $\sqcap_k$-intersection of all $\prec$-minimal models of NF-Mod(P).

Let us list some interesting local properties for $\mathcal{A}, \mathcal{A}' \in NF\text{-}Mod(P)$.

$\mathbf{E}_0(\mathcal{A}, \mathcal{A}')$: $\mathcal{A} \leq_k \mathcal{A}'$,

$\mathbf{E}_1(\mathcal{A}, \mathcal{A}')$: $\exists x \in B_P (\forall y \in B_P : x$ does not depend neg. on y, and $\mathcal{A} \models \neg x, \mathcal{A}' \not\models \neg x)$,

$\mathbf{E}_2(\mathcal{A}, \mathcal{A}')$: $\mathcal{A}' \leq_k \mathcal{A}$.

Theorem 7.3
Let $\mathcal{M}_{NF}$ be given.
a) *Any semantics SEM is a $\leq_k$-extension of Fittings $comp_3$ semantics: for all P, $comp_3(P) \leq_k SEM(P)$.*

b) *Any semantics SEM that satisfies at least "$E_1(\mathcal{A}, \mathcal{A}') \Longrightarrow \mathcal{A} \prec_P \mathcal{A}'$" coincides for stratified programs with the supported semantics M_P^{supp} and is a $\leq_k$-extension of the wellfounded semantics WFS: for all P, $WFS(P) \leq_k SEM(P)$.*

c) *Any semantics SEM satisfying "$E_1(\mathcal{A}, \mathcal{A}')$ or $E_2(\mathcal{A}, \mathcal{A}') \Longrightarrow \mathcal{A} \prec_P \mathcal{A}'$" is a $\leq_k$-extension of WFS': for all P, $WFS'(P) \leq_k SEM(P)$.*

Any of these local properties E_i induces a semantics SEM^{E_i} by defining a relation $\prec_{E_i}$: just interpret the "$\Longrightarrow$" in the previous theorem as "$\Longleftrightarrow$". The framework based on E_0 is still very weak, it corresponds to 3-valued completion $comp_3$:

Theorem 7.4 (Representations)
a) *The semantics SEM^{E_0} induced by E_0 coincides with $comp_3$.*

b) *The semantics SEM^{E_1} induced by E_1 coincides with WFS.*

c) *The semantics SEM^{E_2} induced by E_2 coincides with WFS'.*

The stable semantics can also be represented in our framework: we only have to choose $\prec_P$ in such a way, that the $\prec_P$ minimal models are exactly the stable models of P.

We claim that the following conjecture is true:

Conjecture 7.5 (Charact. of WFS, WFS')
Besides WFS and WFS', there are no other rational and reasonable (in the sense of Definition 6.5) semantics SEM that are induced by some $\mathcal{M}_{NF}$.

8 Disjunctive Logic Programs

While there is only one canonical semantics for positive programs without disjunctions, the least Herbrand model, the situation changes when disjunctive heads are allowed: we can interpret "$\vee$"

- *exclusive*, this leads to GCWA, or
- *inclusive*, this leads to WGCWA (or, equivalently, to the DDR-rule).

In both cases, it is an interesting question and a field of active research to ask for extensions of the respective semantics to larger classes of programs, eg, to the class of *stratified* disjunctive programs or, finally, to the whole class of *all* disjunctive programs. Recently, various semantics have been proposed:

- the "WGCWA" (or, equivalently, the DDR-rule as introduced in [RT88]), that interprets $\vee$ *inclusive* ([RLM89]), and the "GCWA" ([Min82]) that interprets $\vee$ *exclusive*, for *positive* disjunctive programs,
- the *perfect* "PERFECT" and "GCWAS" (see [Prz88b], and [MLR91]) for *stratified* disjunctive programs,

and some semantics defined for general disjunctive programs:

- the *stationary* semantics "STATIONARY" of Przymusinski ([Prz91]),
- the "GDWFS" and the "WF^3" of Baral, Lobo and Minker ([BLM90a] and [BLM91]).

These semantics for disjunctive programs are considered in [Dix92b]. It is straightforward to adapt the formulations of our conditions to the case of disjunctive programs. Obviously the conditions C1 - C4 also have to be modified. We will elaborate on this in a forthcoming paper.

9 Conclusions and Outlook

We have presented, to the best of our knowledge for the first time, a semantical framework for defining and reasoning about semantics of logic programs. This makes it possible to ask for all instances satisfying certain

properties and allows us to prove representation theorems of the kind stated in the last section.

We are currently extending our framework to the case of disjunctive logic programs and hope to get similar representation theorems.

Recently, Pereira et al. ([PAA92]) defined an extension of WFS by adding suitable negative literals. They argue, that from

$$a \leftarrow \neg a$$
$$c \leftarrow \neg a$$

$\neg c$ should follow. They also derive $\{\neg p, q\}$ from

$$a \leftarrow \neg a$$
$$p \leftarrow \neg a$$
$$q \leftarrow \neg p$$

which shows that their semantics is not rational: $\neg a$ is not derivable from P, $\neg p$ is derivable from P but yet p is derivable from $P \cup \{a\}$ (and therefore $\neg p$ is not derivable, as Rationality would require). Their semantics also does not satisfy C3 and, indeed, is not even a model of P: c is false but all rule bodies of c are undefined.

The author learned from Pereira et al.'s work only after this paper was already finished and so could not incorporate their semantics and determine its properties.

Acknowledgements

Many thanks to Martin Müller for lots of useful comments and many fruitful discussions. The comments of some anonymous referees also helped to improve the paper.

References

[ABW88] K. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In Jack Minker, editor, *Foundations of Deductive Databases*, chapter 2, pages 89–148. Morgan Kaufmann, 1988.

[BF89] Nicole Bidoit and Christine Froidevaux. General logical Databases and Programs: Default Logic Semantics and Stratification. *Information and Computation*, to appear, 1989.

[BLM90a] Chitta Baral, Jorge Lobo, and Jack Minker. Generalized Disjunctive Well-founded Semantics for Logic Programs. CS-TR 2436, Computer Science Dept., Univ. Maryland, University of Maryland College Park, Maryland 20742, USA, March 1990.

[BLM90b] Chitta Baral, Jorge Lobo, and Jack Minker. Generalized Well-founded Semantics for Logic Programs. In M. E. Stickel, editor, *10th International Conference on Automated Deduction, LNAI 449, subseries LNCS*, pages 102–116. Springer, J. Siekmann, July 1990.

[BLM91] C. Baral, J. Lobo, and J. Minker. WF^3: A Semantics for Negation in Normal Disjunctive Logic Programs. In Z.W. Ras and M. Zemankova, editors, *Methodologies for Intelligent Systems*, pages 459–468. Springer, Lecture Notes in Artificial Intelligence 542, 1991.

[Dix91a] Jürgen Dix. Classifying Semantics of Logic Programs. In Anil Nerode, Wiktor Marek, and V. S. Subrahmanian, editors, *Logic Programming and Non-Monotonic Reasoning, Proceedings of the first International Workshop*, pages 166–180. Washington D.C, MIT Press, July 1991.

[Dix91b] Jürgen Dix. Cumulativity and Rationality in Semantics of Normal Logic Programs. In J. Dix, K. P. Jantke, and P. H. Schmitt, editors, *Proceedings of the first Workshop on Nonmonotonic and Inductive Logic 1990 in Karlsruhe*, pages 13–37. LNCS 543, Springer, October 1991.

[Dix92a] Jürgen Dix. A Framework for Representing and Characterizing Semantics of Logic Programs. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR92)*. San Mateo, CA, Morgan Kaufmann, 1992.

[Dix92b] Jürgen Dix. Classifying Semantics of Disjunctive Logic Programs. In *LOGIC PROGRAMMING: Proceedings of the 1992 Joint International Conference and Symposium*. MIT Press, November 1992.

[GL88] Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*. MIT Press, 1988.

[KLM90] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. Nonmonotonic Reasoning, Preferential Models and Cumulative Logics. *Artificial Intelligence*, 44(1):167–207, 1990.

[Llo87] John Lloyd. *Foundations of Logic Programming*. Springer- Verlag, 2nd edition, 1987.

[Mak89] David Makinson. General Theory of Cumulative Inference. In Reinfrank, de Kleer, Ginsberg, and Sandewall, editors, *Non-Monotonic Reasoning*. Lecture Notes in Artificial Intelligence 346, Springer-Verlag, January 1989.

[Min82] Jack Minker. On indefinite databases and the closed world assumption. In *Proceedings of*

the 6th Conference on Automated Deduction, New York, pages 292–308. Springer, 1982.

[MLR91] Jack Minker, Jorge Lobo, and Arcot Rajasekar. Circumscription and Disjunctive Logic Programming. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation. Papers in Honor of John McCarthy*, pages 281–304. Academic Press, 1991.

[PAA92] L.M. Pereira, J.J. Alfreres, and J.N. Aparicio. Adding Closed World Assumptions to Well Founded Semantics. In ICOT, editor, *Fifth Generation Computer Systems, Proceedings of the Conference*, pages 562–569. OMSHA, June 1992.

[Prz88a] Teodor Przymusinski. Non-monotonic Reasoning vs. Logic Programming: A new Perspective. In Partridge/Wilks, editor, *Formal Foundations of Artificial Intelligence*. Cambridge University Press, 1988.

[Prz88b] Teodor Przymusinski. On the declarative semantics of deductive databases and logic programs. In Jack Minker, editor, *Foundations of Deductive Databases*, chapter 5, pages 193–216. Morgan Kaufmann, 1988.

[Prz91] Teodor Przymusinski. Stationary Semantics for Normal and Disjunctive Logic Programs. In C. Delobel, M. Kifer, and Y. Masunaga, editors, *DOOD '91, Proceedings of the 2nd International Conference*. Muenchen, Springer, LNCS 566, December 1991.

[RLM89] Arcot Rajasekar, Jorge Lobo, and Jack Minker. Weak Generalized Closed World Assumption. *Journal of Automated Reasoning*, 5:293–307, 1989.

[RT88] Kenneth A. Ross and Rodney A. Topor. Inferring negative Information from disjunctive Databases. *Journal of Automated Reasoning*, 4:397–424, 1988.

[vRS88] Allen van Gelder, Kenneth Ross, and J.S. Schlipf. Unfounded Sets and well-founded Semantics for general logic Programs. In *Proceedings 7th Symposion on Principles of Database Systems*, pages 221–230, 1988.

Answer Sets in General Nonmonotonic Reasoning (Preliminary Report)*

Vladimir Lifschitz
Departments of Computer Sciences and Philosophy
University of Texas at Austin
Austin, TX 78712

Thomas Y.C. Woo
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712

Abstract

Languages of declarative logic programming differ from other modal nonmonotonic formalisms by lack of syntactic uniformity. For instance, negation as failure can be used in the body of a rule, but not in the head; in disjunctive programs, disjunction is used in the head of a rule, but not in the body; in extended programs, negation as failure can be used on top of classical negation, but not the other way around. We argue that this lack of uniformity should not be viewed as a distinguishing feature of logic programming in general. As a starting point, we take a translation from the language of disjunctive programs with negation as failure and classical negation into MBNF—the logic of minimal belief and negation as failure. A class of theories based on this logic is defined, *theories with protected literals*, which is syntactically uniform and contains the translations of all programs. We show that theories with protected literals have a semantics similar to the answer set semantics used in logic programming, and investigate the expressiveness of these theories.

1 Introduction

Investigations on the semantics of negation as failure have shown that declarative languages of logic programming are closely related to the nonmonotonic formalisms developed in Artificial Intelligence. It is known, for instance, that general logic programs can be reduced to default theories in the sense of [Reiter, 1980] by identifying a rule

$$A_0 \leftarrow A_1, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n \qquad (1)$$

with the default

$$A_1 \wedge \ldots \wedge A_m : \neg A_{m+1}, \ldots, \neg A_n \,/\, A_0$$

[Bidoit and Froidevaux, 1987]. Alternatively, general logic programs can be viewed as a special case of autoepistemic theories [Gelfond, 1987].

These ideas have a profound significance for the theory of knowledge representation. They show that representing knowledge in declarative logic programming is very similar to representing knowledge in default logic or in modal nonmonotonic logics. A particularly striking example can be found in research on the frame problem. It is known that expressing temporal persistence by the formula

$$\mathit{Holds}(f,s) \wedge \neg \mathit{Ab}(f,a,s) \supset \mathit{Holds}(f, \mathit{Result}(a,s))$$

leads to difficulties [Hanks and McDermott, 1987]. This fact prompted several authors ([Eshghi and Kowalski, 1989], [Evans, 1989], [Apt and Bezem, 1990]) to experiment with the corresponding logic programming rule

$$\mathit{Holds}(f, \mathit{Result}(a,s)) \leftarrow \mathit{Holds}(f,s), \mathit{not}\ \mathit{Ab}(f,a,s). \qquad (2)$$

Independently, Morris [1988] proposed to express the same principle of reasoning by the default

$$\mathit{Holds}(f,s) : \neg \mathit{Ab}(f,a,s) \,/\, \mathit{Holds}(f, \mathit{Result}(a,s)).$$

It is clear that this default is the counterpart of (2) under the Bidoit/Froidevaux translation.

Although general logic programs are a special case of default theories, it is not true that declarative logic programming as a whole is merely a subset of default logic. This can be demonstrated on the example of

*This research was supported in part by NSF grants NCR-9004464 and IRI-9101078.

"disjunctive logic programs." Gelfond and Lifschitz [1991] discuss disjunctive rules of the form

$$L_1 \mid \ldots \mid L_l \leftarrow L_{l+1}, \ldots, L_m, \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_n, \tag{3}$$

($n \geq m \geq l \geq 0$), where each L_i is a literal (an atom possibly preceded by $\neg$). It is not clear how to represent such a rule by a default or by a set of defaults; apparently, default logic and the language of disjunctive programs only partially overlap. *Disjunctive default logic* [Gelfond *et al.*, 1991] was proposed as a nonmonotonic formalism which can serve as a common extension of these two languages.

The logic of *minimal belief and negation as failure (MBNF)*[1] is another such formalism. It uses two independent nonmonotonic modalities: the minimal belief operator B and the negation as failure operator *not*. A default

$$\alpha : \beta_1, \ldots, \beta_m \,/\, \gamma$$

is represented in this language by the formula[2]

$$\mathrm{B}\alpha \wedge \mathit{not}\neg\beta_1 \wedge \ldots \wedge \mathit{not}\neg\beta_m \supset \mathrm{B}\gamma. \tag{4}$$

A disjunctive rule (3) can be identified with the formula

$$\mathrm{B}L_{l+1} \wedge \ldots \wedge \mathrm{B}L_m \wedge \mathit{not}\ L_{m+1} \wedge \ldots \wedge \mathit{not}\ L_n \supset \mathrm{B}L_1 \vee \ldots \vee \mathrm{B}L_l. \tag{5}$$

It is also possible to embed a rather general form of circumscription into MBNF [Lifschitz, 1992]; circumscribing a predicate P is expressed by the axiom

$$\forall x(\mathit{not}\ P(x) \supset \mathrm{B}\neg P(x)).$$

These facts show that the logic of minimal belief and negation as failure is very expressive. We should note, however, that the interesting concept of "strong introspection" [Gelfond, 1991] is apparently not expressible in it.

The embedding of logic programs into MBNF stresses the epistemic character of the "connectives" $\leftarrow$ and $\mid$. The rule $L_1 \leftarrow L_2$ is different from the implication $L_2 \supset L_1$; it is rendered by the combination $\mathrm{B}L_2 \supset \mathrm{B}L_1$, which includes both the classical connective $\supset$ and the epistemic operator B. The rule $L_1 \mid L_2 \leftarrow$ is different from the disjunction $L_1 \vee L_2$; it is rendered by the epistemic combination $\mathrm{B}L_1 \vee \mathrm{B}L_2$.

Another reason why this embedding may be of interest is related to the fact that the syntax of MBNF is *uniform*. Propositional connectives and the epistemic operators B, *not* can be applied in formulas of MBNF any number of times and in any order. The syntax of rules (3) is, in this sense, different. Each rule contains only one occurrence of $\leftarrow$; we are not allowed to form a "nested" rule by applying $\leftarrow$ to two rules formed earlier. Epistemic disjunction is allowed in the head of a rule, but not in the body. Negation as failure can be used in the body, but not in the head. Classical negation can be applied to atoms only—never on top of *not*, $\mid$ or $\leftarrow$.

In this paper we argue that this lack of uniformity is not an essential feature of logic programming. We define a class of formulas of MBNF that includes all formulas (5) and is syntactically rather uniform, and show that such formulas are, in a sense, "semantically similar" to logic programming rules. If all axioms of a theory T belong to this class, we call T a "theory with protected literals," or a "PL-theory." We hope that the study of PL-theories will help us better understand the place of declarative logic programming among nonmonotonic formalisms in general.

At this stage, we restrict our attention to the propositional case.

The semantics of rules (3) is defined in [Gelfond and Lifschitz, 1991] in terms of "answer sets." An answer set of a program is a set of literals. The semantics of MBNF is defined in terms of Kripke-style sets of "possible worlds." The relationship between the two systems is described in [Lifschitz, 1992] by establishing a simple correspondence ω between sets of literals and sets of possible worlds. If the axioms of a theory have the form (5), then the sets of worlds that appear in its models have the form $\omega(\Sigma)$, where Σ is an answer set of the corresponding program.

We show that ω serves as a correspondence between the answer sets and the models not only for disjunctive programs, but for all PL-theories. This theorem suggests that theories of this type, in spite of the relatively general syntactic form of their axioms, can be viewed as logic programs.

Next we want to compare the expressive power of

[1] See [Lifschitz, 1992]. This is a modified version of the system described in the preliminary report [Lifschitz, 1991]. The description of the propositional fragment is reproduced in Section 2 below. The system is a modification and extension of the "logic of grounded knowledge" introduced by Lin and Shoham [1990]. The concept of minimal belief (or "minimal knowledge," or "maximal ignorance") was formalized earlier, in various ways, by several authors, including Konolige [1982], Halpern and Moses [1984], Shoham [1986] and Lin [1988].

[2] In the propositional case, this formula represents the meaning of the default as originally defined by Reiter [1980]. In the presence of variables, it corresponds to the modification of default logic proposed in [Lifschitz, 1990].

arbitrary PL-theories with the expressive power of disjunctive programs. Two theories are said to be equivalent if they have the same models; in the case of PL-theories, we can alternatively say, "the same answer sets." Given a PL-theory, can we always find an equivalent disjunctive program?

The answer to this question is no, because of an interesting syntactic feature that one can find in a PL-theory: The operator *not* may occur in the axioms positively. (It is clear that all occurrences of *not* in a disjunctive rule (5) are negative.) We give examples of PL-theories that are not equivalent to disjunctive programs.

2 Propositional MBNF

The review of the propositional fragment of MBNF below follows [Lifschitz, 1992], except that, in this presentation, the language is assumed to include the propositional constant T ("true"). We start with a set of propositional symbols, *atoms*, which includes T. Formulas are built from atoms using the propositional connectives $\neg$ and $\wedge$ and the modal operators B and *not*. The other connectives are defined in terms of $\neg$ and $\wedge$ in the usual way; F ("false") stands for the literal $\neg\mathsf{T}$. A *theory* is a set of formulas (*axioms*).

If a formula or a theory does not contain the operator *not*, we call it *positive*. This terminology is suggested by the use of the word "positive" in logic programming, and it is not related to the distinction between positive and negative occurrences, familiar from classical logic. In MBNF, the sign of an occurrence of a symbol in a formula can be defined as follows: An occurrence is *positive* if it is in the range of an even number of $\neg$'s and *not*'s, and *negative* otherwise.

An *interpretation* is a set I of atoms such that $\mathsf{T} \in I$. A *structure* is a pair (I, S), where I is an interpretation, and S a set of interpretations.

The relation $<$ between structures is defined as follows: $(I, S) < (I', S')$ if S is a proper subset of S'. The maximality of a structure relative to this relation expresses the idea of "minimal belief": The larger the set of "possible worlds" is, the fewer propositions are believed.

We define when a positive formula F is *true* in a structure (I, S), as follows.

- If F is an atom, F is true in (I, S) iff $F \in I$.
- $\neg F$ is true in (I, S) iff F is not true in (I, S).
- $F \wedge G$ is true in (I, S) iff F and G are both true in (I, S).
- $\mathrm{B}F$ is true in (I, S) iff, for every $J \in S$, F is true in (J, S).

A *model* of a positive theory T is any structure maximal among those in which the axioms of T are true. For instance, the models of $\{\mathrm{B}p\}$, where p is an atom, are the structures of the form $(I, \{J : p \in J\})$, where I is any interpretation. The models of $\{\mathrm{B}\neg p\}$ have the form $(I, \{J : p \notin J\})$. The models of $\{\mathrm{B}p \vee \mathrm{B}q\}$ have the forms $(I, \{J : p \in J\})$ and $(I, \{J : q \in J\})$. The models of $\{\mathrm{B}(p \vee q)\}$ have the form $(I, \{J : p \in J \text{ or } q \in J\})$.

A positive formula F is a *theorem* of a positive theory T if F is true in every model of T. This relation is nonmonotonic. For instance, $\neg\mathrm{B}q$ is a theorem of $\{\mathrm{B}p\}$, but not a theorem of $\{\mathrm{B}p, \mathrm{B}q\}$.

In order to extend the definition of a model to nonpositive theories, we first need to extend the definition of truth to nonpositive formulas. In the presence of both B and *not*, truth will be defined relative to a triple $(I, S^{\mathfrak{b}}, S^{\mathfrak{n}})$, where $S^{\mathfrak{b}}$ and $S^{\mathfrak{n}}$ are sets of interpretations; $S^{\mathfrak{b}}$ serves as the set of "possible worlds" for the purpose of defining the meaning of B, and $S^{\mathfrak{n}}$ plays the same role for the operator *not*.

For an interpretation I and two sets of interpretations $S^{\mathfrak{b}}$, $S^{\mathfrak{n}}$, we define when a formula F is *true* in $(I, S^{\mathfrak{b}}, S^{\mathfrak{n}})$, as follows.

- If F is an atom, F is true in $(I, S^{\mathfrak{b}}, S^{\mathfrak{n}})$ iff $F \in I$.
- $\neg F$ is true in $(I, S^{\mathfrak{b}}, S^{\mathfrak{n}})$ iff F is not true in $(I, S^{\mathfrak{b}}, S^{\mathfrak{n}})$.
- $F \wedge G$ is true in $(I, S^{\mathfrak{b}}, S^{\mathfrak{n}})$ iff F and G are both true in $(I, S^{\mathfrak{b}}, S^{\mathfrak{n}})$.
- $\mathrm{B}F$ is true in $(I, S^{\mathfrak{b}}, S^{\mathfrak{n}})$ iff, for every $J \in S^{\mathfrak{b}}$, F is true in $(J, S^{\mathfrak{b}}, S^{\mathfrak{n}})$.
- *not* F is true in $(I, S^{\mathfrak{b}}, S^{\mathfrak{n}})$ iff, for some $J \in S^{\mathfrak{n}}$, F is not true in $(J, S^{\mathfrak{b}}, S^{\mathfrak{n}})$.

This definition is a generalization of the definition of truth for positive formulas, in the sense that a positive formula is true in $(I, S^{\mathfrak{b}}, S^{\mathfrak{n}})$ iff it is true in $(I, S^{\mathfrak{b}})$.

For any theory T and any set of interpretations S, by $\Gamma(T, S)$ we denote the set of all maximal structures (I, S') such that the axioms of T are true in (I, S', S). Intuitively, $\Gamma(T, S)$ consists of the structures that can be considered the models of T provided that the negation as failure operator is interpreted relative to the set of possible worlds S.

A structure (I, S) is a *model* of T if $(I, S) \in \Gamma(T, S)$. It is easy to check, for instance, that the models of $\{not\ p \supset \mathrm{B}q\}$ are the structures of the form $(I, \{J : q \in J\})$. For positive theories, this definition is equivalent to the one given before. The reader is referred to [Lifschitz, 1992] for further examples.

3 Answer Sets

Let Σ be a set of literals. By $\Sigma^{\mathfrak{p}}$ we denote the set of atoms that belong to Σ, and by $\Sigma^{\mathfrak{n}}$ the set of atoms

whose negations belong to Σ, so that

$$\Sigma = \Sigma^p \cup \{\neg A : A \in \Sigma^n\}.$$

Furthermore, $\omega(\Sigma)$ stands for the set of interpretations I such that $\Sigma^p \subset I$ and $\Sigma^n \cap I = \emptyset$. For example,

$$\omega(\{p, \neg q\}) = \omega(\{\mathsf{T}, p, \neg q\}) = \{I : p \in I, q \notin I\},$$

$$\omega(\{p, \neg p\}) = \omega(\{\mathsf{F}\}) = \emptyset.$$

We want to define, for theories T of a possibly more general form, when a set of literals Σ is an "answer set" of T, in such a way that the models of T will be the structures of the form $(I, \omega(\Sigma))$. For example, this can be done for the theory $\{\mathrm{B}p\}$ by declaring $\{p\}$ (or $\{\mathsf{T}, p\}$) to be its only answer set. It can be also done for $\{\mathrm{B}\neg p\}$, $\{\mathrm{B}p \vee \mathrm{B}q\}$ and $\{not\ p \supset \mathrm{B}q\}$, but not for $\{\mathrm{B}(p \vee q)\}$.

What is different about the last axiom is that a connective is applied in it to two atoms directly, without first "protecting" them by a modal operator. This observation suggests the following definitions.

Protected literals are formulas of the forms $\mathrm{B}L$ and *not* L, where L is a literal, and the atom T. (Including T in this definition is convenient, but not essential.) A formula F is a *formula with protected literals*, or a *PL-formula*, if each occurrence of an atom in F is a part of a protected literal. Alternatively, PL-formulas can be characterized as the formulas that can be built from protected literals using $\neg$, B, *not* and $\wedge$. For instance, every formula of the form (5)—and, more generally, every propositional combination of protected literals—is a PL-formula. Clearly, *not* $\mathrm{B}\neg p$ is a PL-formula also; p and $\mathrm{B}(p \vee q)$ are not PL-formulas.

A *theory with protected literals*, or a *PL-theory*, is a theory whose axioms are PL-formulas. We will define the concept of an answer set for arbitrary PL-theories. This will be done in three steps. First, we will consider the theories whose axioms are propositional combinations of positive protected literals (that is, of protected literals that do not contain *not*). This class covers the translations of positive disjunctive programs. Then the definition will be extended to the combinations of arbitrary protected literals. This class covers the translations of all disjunctive programs. Finally, the definition will be extended to arbitrary PL-theories.

By *Lit* we denote the set of all literals. A set $\Sigma \subset Lit$ is *closed* if it satisfies two conditions:

- $\mathsf{T} \in \Sigma$.
- If Σ contains a pair of complimentary literals, then $\Sigma = Lit$.

Note that, for any closed set of literals Σ, $\mathsf{F} \in \Sigma$ if and only if $\Sigma = Lit$.

The *satisfaction relation* between a set Σ of literals and a propositional combination F of positive protected literals is defined inductively, as follows:

- $\Sigma \models \mathsf{T}$.
- $\Sigma \models \mathrm{B}L$ iff $L \in \Sigma$.
- $\Sigma \models \neg F$ iff $\Sigma \not\models F$.
- $\Sigma \models F \wedge G$ iff $\Sigma \models F$ and $\Sigma \models G$.

The Definition of Answer Sets, Step 1. Let T be a theory whose axioms are propositional combinations of positive protected literals. A set Σ of literals is an *answer set* of T if it is a minimal (relative to set inclusion) closed set such that, for every axiom F of T, $\Sigma \models F$.

For instance, the only answer set of $\{\mathrm{B}p\}$ is $\{\mathsf{T}, p\}$; the only answer set of $\{\mathrm{B}p \supset \mathrm{B}q\}$ is $\{\mathsf{T}\}$; the answer sets of $\{\mathrm{B}p \vee \mathrm{B}q\}$ are $\{\mathsf{T}, p\}$ and $\{\mathsf{T}, q\}$. The only answer set of $\{\mathsf{BF}\}$ is Lit; $\{\mathsf{F}\}$ has no answer sets.

For any propositional combination F of protected literals and any set Σ of literals, the *reduct of F relative to Σ* is the formula F^Σ obtained by replacing each subformula of the form *not* L in F by F if $L \in \Sigma$, and by T otherwise. For instance, if F is *not* $p \supset \mathrm{B}q$, then $F^\emptyset$ is $\mathsf{T} \supset \mathrm{B}q$, and $F^{\{p\}}$ is $\mathsf{F} \supset \mathrm{B}q$. This is a generalization of the procedure used in the definition of "stable models" [Gelfond and Lifschitz, 1988].

If the axioms of T are propositional combinations of protected literals, then T^Σ stands for $\{F^\Sigma : F \in T\}$. For theories of the form T^Σ, the notion of an answer set was defined in Step 1.

The Definition of Answer Sets, Step 2. Let T be a theory whose axioms are propositional combinations of protected literals. A set Σ of literals is an *answer set* of T if it is an answer set of T^Σ.

It is easy to check, for instance, that the only answer set of $\{not\ p \supset \mathrm{B}q\}$ is $\{\mathsf{T}, q\}$. The theories $\{not\ p \supset \mathrm{B}p\}$, $\{\neg not\ p\}$ and $\{not\ \mathsf{T}\}$ have no answer sets; the only answer set of $\{not\ \mathsf{F}\}$ is $\{\mathsf{T}\}$.

For a theory corresponding to a set of disjunctive rules (3), this definition is essentially equivalent to the one given in [Gelfond and Lifschitz, 1991]; the only difference is that an answer set as defined here includes T, and, if it is inconsistent, also F.

For any PL-formula F, let F^* be the propositional combination of protected literals defined inductively, as follows:

- F^* is F, if F is a protected literal.

- $(\neg F)^*$ is $\neg F^*$.
- $(F \wedge G)^*$ is $F^* \wedge G^*$.
- $(\mathrm{B}F)^*$ is $F^* \vee \mathrm{BF}$, if F is not a literal.
- $(\mathit{not}\ F)^*$ is $\neg F^* \wedge \mathit{not}\ \mathsf{F}$, if F is not a literal.

For any PL-theory T, T^* stands for $\{F^* : F \in T\}$. For theories of the form T^*, the notion of an answer set was defined in Step 2.

The Definition of Answer Sets, Step 3. Let T be a PL-theory. A set Σ of literals is an *answer set* of T if it is an answer set of T^*.

Take, for instance, $T = \{\mathrm{B}(\mathrm{B}p \wedge \neg \mathrm{B}p)\}$. The definition tells us that T has the same answer sets as

$$\{(\mathrm{B}p \wedge \neg \mathrm{B}p) \vee \mathrm{BF}\}. \qquad (6)$$

Consequently, the only answer set of T is *Lit*. We see that T is not equivalent to the theory $\{\mathrm{B}p \wedge \neg \mathrm{B}p\}$—the latter has no answer sets.

The following theorem shows that we have achieved the goal stated at the beginning of this section.

Theorem 1. *A structure (I, S) is a model of a PL-theory T if and only if $S = \omega(\Sigma)$ for some answer set Σ of T.*

The proofs of theorems are given in the appendix.

4 Disjunctive Rules

A *disjunctive rule* is a formula of the form (5) in which none of the literals L_i is T or F. We would like to compare the expressiveness of the theories whose axioms are disjunctive rules with the expressiveness of arbitrary PL-theories.

The definition of answer sets for arbitrary PL-theories (Step 3) reduces the axioms to propositional combinations of protected literals by a simple transformation, which does not change significantly the syntactic structure or the size of the formula. Consequently, without loss of generality, we can restrict our attention to the theories whose axioms are propositional combinations of protected literals.

Furthermore, a propositional combination of protected literals can be replaced by its "conjunctive normal form"—a set of disjunctions of protected literals and their negations. For instance,

$$((\mathrm{B}p \vee \mathit{not}\ q) \wedge \mathit{not}\ r) \supset \mathrm{B}s$$

will turn into the pair of disjunctions which can be written as

$$(\mathrm{B}p \wedge \mathit{not}\ r) \supset \mathrm{B}s,$$
$$(\mathit{not}\ q \wedge \mathit{not}\ r) \supset \mathrm{B}s.$$

This is similar to the transformation of logic programs proposed in [Lloyd and Topor, 1984]. Note, however, that this reduction may lead to the exponential growth of the axiom set.

A disjunction of protected literals and their negations can be written as a disjunctive rule (5) if

- it contains no positive occurrences of *not*, and
- it contains no occurrences of BT, BF, *not* T, *not* F.

The following theorems show that the second restriction is inessential.

Theorem 2. *Let T be a theory whose axioms are propositional combinations of protected literals, and let T' be obtained from it by substituting T for all occurrences of BT, and F for all occurrences of not T, in every axiom. Then T and T' have the same answer sets.*

Theorem 3. *Let T be a theory whose axioms are propositional combinations of protected literals, and let T' be obtained from it by substituting F for all occurrences of BF, and T for all occurrences of not F, in every axiom. For any set of literals Σ other than Lit, Σ is an answer set of T if and only if it is an answer set of T'.*

The answer set *Lit* may be lost as the result of the last transformation, as can be seen from the examples $\{\mathsf{BF}\}$ and (6).

On the other hand, the first restriction—the absence of positive occurrences of *not*—turns out to be essential. Axiom sets without positive occurrences of *not* have the following property.

Theorem 4. *If the axioms of a PL-theory contain no positive occurrences of not, then it cannot have two answer sets of which one is a proper subset of the other.*

This is a generalization of Lemma 1 from [Gelfond and Lifschitz, 1991]. It is similar to the minimality of extensions property in default logic ([Reiter, 1980], Theorem 2.4).

Corollary. *If the axioms of a PL-theory T contain no positive occurrences of not, and Lit is an answer set of T, then T has no other answer sets.*

This is a generalization of Proposition 1 from [Gelfond and Lifschitz, 1991]. It is similar to Corollary 2.3 from [Reiter, 1980].

Using Theorem 4, we can show that a PL-theory may have a combination of answer sets that would be impossible without positive occurrences of *not* in the axioms. For instance, the theory

$$\{\mathrm{B}p \vee \mathit{not}\ p\} \tag{7}$$

has two answer sets, $\{\mathrm{T}\}$ and $\{\mathrm{T}, p\}$. By Theorem 4, it is not equivalent to any set of disjunctive rules. Moreover, a PL-theory can have *Lit* as one of several answer sets. For instance, the answer sets of the theory

$$\{\mathrm{B}p \vee \mathit{not}\ p,\ \mathrm{B}\neg p \vee \mathit{not}\neg p\}$$

are $\{\mathrm{T}\}$, $\{\mathrm{T}, p\}$, $\{\mathrm{T}, \neg p\}$ and *Lit*.

5 Discussion

1. Where is the line separating the languages of "declarative logic programming" from other modal nonmonotonic formalisms? Our view on what is essential about logic programming is that its semantics can be described in terms of *sets of literals*—objects that are much simpler than Kripke models.

Another possible view is that a logic programming language, unlike nonmonotonic formalisms of other kinds, always comes equipped with a standard query evaluation method; it has a procedural semantics, in addition to the declarative one. From this perspective, sets of rules of the form (1) can be counted as logic programs because one can execute them using a Prolog interpreter. But if we intend to use the language for the purpose of representing knowledge, and not for programming, then there is no reason to ascribe any special role to the Prolog search strategy. Much work has been done on alternative query evaluation methods, such as the "magic set" method of [Bancilhon *et al.*, 1986], which may produce an answer when Prolog would not terminate. The development of better query evaluation procedures in logic programming is similar to the development of more powerful theorem provers for other kinds of nonmonotonic formalisms. The existence of incomplete, but useful query evaluation procedures is not a distinguishing feature of logic programming.

2. The use of (some syntactic variant of) nested combinations of protected literals may give representational advantages over a "flat" syntax, similar to the advantages of the extension of Prolog described by Lloyd and Topor [1984]. There are several differences between our proposal and theirs. First, the answer set semantics has grown from the use of minimal models by van Emden and Kowalski [1976] and from the generalizations of this idea in [Apt *et al.*, 1988], [Przymusinski, 1988], [Gelfond and Lifschitz, 1988], rather than from the completion semantics of [Clark, 1978]. Second, we include rules with disjunctive heads. Third, we distinguish between negation as failure and classical negation.

3. A disjunction of protected literals and their negations can be written in the form

$$\mathrm{B}L_{l+1} \wedge \ldots \wedge \mathrm{B}L_m \wedge \mathit{not}\ L_{m+1} \wedge \ldots \wedge \mathit{not}\ L_n \supset \mathrm{B}L_1 \vee \ldots \vee \mathrm{B}L_k \vee \mathit{not}\ L_{k+1} \vee \ldots \vee \mathit{not}\ L_l,$$

or, in "logic programming notation," as

$$L_1 \mid \ldots \mid L_k \mid \mathit{not}\ L_{k+1} \mid \ldots \mid \mathit{not}\ L_l \leftarrow L_{l+1}, \ldots, L_m, \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_n. \tag{8}$$

For example, (7) is, in this notation,

$$p \mid \mathit{not}\ p \leftarrow .$$

This rule has two answer sets; one of them includes p ("p is true"), and the other includes neither p nor $\neg p$ ("the truth value of p is unknown"). It remains to be seen whether rules like this may have applications to knowledge representation.

4. There is a possibility that including a rule of the form (8) in a program may be computationally advantageous, when this rule is redundant from the point of view of the declarative semantics. Let Π be a set of rules (1) which includes, among others, the positive rule

$$p \leftarrow q, r.$$

If we know that q succeeds, but p fails, then we can conclude that r fails. This reasoning can be formally represented as using the "contrapositive" rule

$$\mathit{not}\ r \leftarrow q, \mathit{not}\ p.$$

We have shown that such rules can be given a declarative semantics. It is easy to prove that adding this rule to Π does not change its answer sets.

This idea was suggested to us by the informal discussion of "contrapositive rules" in [Kowalski and Kim, 1991]. About the rule

$$\mathit{demo}(T, Q) \leftarrow \mathit{demo}(T, \mathit{or}(P, Q)), \mathit{demo}(\mathit{not}(P))$$

Kowalski and Kim observe that it would be useful also "in its contrapositive form"

$$not\ demo(not(P)) \leftarrow demo(T, or(P, Q)), not\ demo(T, Q).$$

"Such use of contrapositives, however, is not possible within currently available logic programming systems." The generalization of answer sets proposed in this paper may provide a theoretical foundation for the use of contrapositives.

Acknowledgements

We are grateful to Michael Gelfond, G. N. Kartha, Norman McCain and Grigori Schwarz for comments on a draft of this paper.

References

[Apt and Bezem, 1990] Krzysztof Apt and Marc Bezem. Acyclic programs. In David Warren and Peter Szeredi, editors, *Logic Programming: Proc. of the Seventh Int'l Conf.*, pages 617–633, 1990.

[Apt *et al.*, 1988] Krzysztof Apt, Howard Blair, and Adrian Walker. Towards a theory of declarative knowledge. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, San Mateo, CA, 1988.

[Bancilhon *et al.*, 1986] Francois Bancilhon, David Maier, Yehoshua Sagiv, and Jeffrey Ullman. Magic sets and other strange ways to implement logic programs. In *Proc. of the Fifth Symp. on Principles of Database Systems*, pages 1–15, 1986.

[Bidoit and Froidevaux, 1987] Nicole Bidoit and Christine Froidevaux. Minimalism subsumes default logic and circumscription. In *Proc. of LICS-87*, pages 89–97, 1987.

[Clark, 1978] Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[Emden and Kowalski, 1976] Maarten van Emden and Robert Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.

[Eshghi and Kowalski, 1989] Kave Eshghi and Robert Kowalski. Abduction compared with negation as failure. In Giorgio Levi and Maurizio Martelli, editors, *Logic Programming: Proc. of the Sixth Int'l Conf.*, pages 234–255, 1989.

[Evans, 1989] Chris Evans. Negation-as-failure as an approach to the Hanks and McDermott problem. In *Proc. of the Second Int'l Symp. on Artificial Intelligence*, 1989.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Logic Programming: Proc. of the Fifth Int'l Conf. and Symp.*, pages 1070–1080, 1988.

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[Gelfond *et al.*, 1991] Michael Gelfond, Vladimir Lifschitz, Halina Przymusińska, and Miroslaw Truszczyński. Disjunctive defaults. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proc. of the Second Int'l Conf.*, pages 230–237, 1991.

[Gelfond, 1987] Michael Gelfond. On stratified autoepistemic theories. In *Proc. AAAI-87*, pages 207–211, 1987.

[Gelfond, 1991] Michael Gelfond. Strong introspection. In *Proc. AAAI-91*, 1991.

[Halpern and Moses, 1984] Joseph Halpern and Yoram Moses. Towards a theory of knowledge and ignorance: preliminary report. Technical Report RJ 4448 (48136), IBM, 1984.

[Hanks and McDermott, 1987] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, 1987.

[Konolige, 1982] Kurt Konolige. Circumscriptive ignorance. In *Proc. of AAAI-82*, pages 202–204, 1982.

[Kowalski and Kim, 1991] Robert Kowalski and Jin-Sang Kim. A metalogic programming approach to multi-agent knowledge and belief. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 231–246. Academic Press, 1991.

[Lifschitz, 1990] Vladimir Lifschitz. On open defaults. In John Lloyd, editor, *Computational Logic: Symposium Proceedings*, pages 80–95. Springer, 1990.

[Lifschitz, 1991] Vladimir Lifschitz. Nonmonotonic databases and epistemic queries. In *Proc. of IJCAI-91*, 1991.

[Lifschitz, 1992] Vladimir Lifschitz. Minimal belief and negation as failure. Submitted for publication, 1992.

[Lin and Shoham, 1990] Fangzhen Lin and Yoav Shoham. Epistemic semantics for fixed-points non-monotonic logics. In Rohit Parikh, editor, *Theoretical Aspects of Reasoning about Knowledge: Proc. of the Third Conf.*, pages 111–120, 1990.

[Lin, 1988] Fangzhen Lin. Circumscription in a modal logic. In Moshe Vardi, editor, *Theoretical Aspects of Reasoning about Knowledge: Proc. of the Second Conf.*, pages 113–127, 1988.

[Lloyd and Topor, 1984] John Lloyd and Rodney Topor. Making Prolog more expressive. *Journal of Logic Programming*, 3:225–240, 1984.

[Morris, 1988] Paul Morris. The anomalous extension problem in default reasoning. *Artificial Intelligence*, 35(3):383–399, 1988.

[Przymusinski, 1988] Teodor Przymusinski. On the declarative semantics of deductive databases and logic programs. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, San Mateo, CA, 1988.

[Reiter, 1980] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1,2):81–132, 1980.

[Shoham, 1986] Yoav Shoham. Chronological ignorance: Time, nonmonotonicity, necessity and causal theories. In *Proc. of AAAI-86*, pages 389–393, 1986.

Appendix. Proofs of Theorems

A1. Preliminary Lemmas

The following facts are used in the proofs of Theorems 1–4.

Lemma 1 [Lifschitz, 1992]. *Let Σ, Σ' be closed sets of literals.*

(a) $\omega(\Sigma) = \emptyset$ *iff* $\Sigma = Lit$.

(b) $\bigcap_{I \in \omega(\Sigma)} I = \Sigma^p$.

(c) $\bigcup_{I \in \omega(\Sigma)} I = \overline{\Sigma^n}$ *where* $\overline{\Sigma^n}$ *denotes the complement of* Σ^n.

(d) $\Sigma \subset \Sigma'$ *iff* $\omega(\Sigma') \subset \omega(\Sigma)$.

(e) $\Sigma = \Sigma'$ *iff* $\omega(\Sigma) = \omega(\Sigma')$.

□

As defined in [Lifschitz, 1992], two sets of interpretations S_1, S_2 are *equivalent* if

$$\bigcap_{I \in S_1} I = \bigcap_{I \in S_2} I \text{ and } \bigcup_{I \in S_1} I = \bigcup_{I \in S_2} I.$$

Lemma 2. *Let I, J be interpretations, and S, S_1, S_2 sets of interpretations. If S_1 and S_2 are equivalent, then, for any PL-formula F,*

$$F \text{ is true in } (I, S_1, S) \text{ iff } F \text{ is true in } (J, S_2, S).$$

□

This is a generalization of an observation made in [Lifschitz, 1992]. The assumption that F is a PL-formula is essential; a simple counterexample is $\mathrm{B}(p \vee q)$, with $S_1 = \{\{p\}, \{q\}\}$ and $S_2 = \{\{\}, \{p, q\}\}$.

Proof. By structural induction.

(1) Base case. Protected literals.

- T. Straightforward.
- BA, where A is an atom.

  BA is true in (I, S_1, S)
  iff $\forall K \in S_1 : A$ is true in (K, S_1, S)
  iff $\forall K \in S_1 : A \in K$
  iff {set theory}
  $A \in \bigcap_{K \in S_1} K$
  iff {S_1 equivales S_2}
  $A \in \bigcap_{K \in S_2} K$
  iff {set theory}
  $\forall K \in S_2 : A \in K$
  iff $\forall K \in S_2 : A$ is true in (K, S_2, S)
  iff BA is true in (J, S_2, S)
- B$\neg A$, where A is an atom.

  B$\neg A$ is true in (I, S_1, S)
  iff $\forall K \in S_1 : \neg A$ is true in (K, S_1, S)
  iff $\forall K \in S_1 : A$ is not true in (K, S_1, S)
  iff $\forall K \in S_1 : A \notin K$
  iff {set theory}
  $A \notin \bigcup_{K \in S_1} K$
  iff {S_1 equivales S_2}
  $A \notin \bigcup_{K \in S_2} K$
  iff {set theory}
  $\forall K \in S_2 : A \notin K$
  iff $\forall K \in S_2 : A$ is not true in (K, S_2, S)
  iff $\forall K \in S_2 : \neg A$ is true in (K, S_2, S)
  iff B$\neg A$ is true in (J, S_2, S)
- *not* L, where L is a literal.

not L is true in (I, S_1, S)
iff $\exists K \in S : L$ is not true in (K, S_1, S)
iff $\exists K \in S : L \notin K$
iff $\exists K \in S : L$ is not true in (K, S_2, S)
iff *not* L is true in (J, S_2, S)

(2) Induction step. The formula has one of the forms $\neg F$, BF, *not* F or $F \wedge G$, where F and G are PL-formulas. Straightforward. □

Using Lemma 2, it is easy to observe that (I, S) is a model of a PL-theory T if and only if (J, S) is a model of T.

Lemma 3 [Lifschitz, 1992]. *For any set of interpretations S, there exists a closed set of literals Σ such that $\omega(\Sigma)$ contains S and is equivalent to it.*

A2. Proof of Theorem 1

A PL-formula is *basic* if it is a propositional combination of protected literals. For example, *not* $p \supset$ Bp is a basic PL-formula, while B *not* $p \wedge$ Bp is not. A *basic* PL-theory is a PL-theory whose axioms are basic. We prove Theorem 1 via two steps. In step 1 (Part A), we establish the theorem for basic PL-theories. Then in step 2 (Part B), we extend the theorem to all PL-theories using Part A as a lemma.

Lemma 4. *Let I be an interpretation, S a set of interpretations, Σ a closed set of literals, and L a literal. Then*

not L is true in $(I, S, \omega(\Sigma))$ iff $L \notin \Sigma$.

Proof.

Case 1. L is A, where A is an atom.

not A is true in $(I, S, \omega(\Sigma))$
iff $\exists J \in \omega(\Sigma) : A$ is not true in $(J, S, \omega(\Sigma))$
iff $\exists J \in \omega(\Sigma) : A \notin J$
iff {set theory}
$A \notin \bigcap_{J \in \omega(\Sigma)} J$
iff {Lemma 1(b)}
$A \notin \Sigma^p$
iff $A \notin \Sigma$

Case 2. L is $\neg A$, where A is an atom.

not $\neg A$ is true in $(I, S, \omega(\Sigma))$
iff $\exists J \in \omega(\Sigma) : \neg A$ is not true in $(J, S, \omega(\Sigma))$
iff $\exists J \in \omega(\Sigma) : A$ is true in $(J, S, \omega(\Sigma))$
iff $\exists J \in \omega(\Sigma) : A \in J$
iff {set theory}
$A \in \bigcup_{J \in \omega(\Sigma)} J$
iff {Lemma 1(c)}
$A \in \overline{\Sigma^n}$
iff $\neg A \notin \Sigma$

□

Lemma 5. *Let I be an interpretation, S, S' sets of interpretations, Σ a closed set of literals, and F a basic PL-formula. Then*

F^Σ is true in (I, S, S') iff F is true in $(I, S, \omega(\Sigma))$.

Proof. By structural induction.

(1) Base case.

- T and BL.
  These are positive PL-formulas, hence F^Σ is identical to F. It remains to notice that the truth of positive formulas relative to a triple (I, S, S') does not depend on S'.
- *not* L, where L is a literal.
  Recall that

$$(not\ L)^\Sigma \text{ is } \begin{cases} \mathsf{T}, & \text{if } L \notin \Sigma, \\ \mathsf{F}, & \text{if } L \in \Sigma. \end{cases}$$

  Thus

$$(not\ L)^\Sigma \text{ is true in } (I, S, S') \text{ iff } L \notin \Sigma.$$

  By Lemma 4, we obtain
  $(not\ L)^\Sigma$ is true in (I, S, S') iff *not* L is true in $(I, S, \omega(\Sigma))$.

(2) Induction step. The formula has the form $\neg F$ or $F \wedge G$, where F and G are PL-formulas. Straightforward. □

Lemma 6. *Let I be an interpretation, S a set of interpretations, Σ a set of literals, and F a positive basic PL-formula. Then*

$\Sigma \models F$ iff F is true in $(I, \omega(\Sigma), S)$.

Proof. By structural induction.

(1) Base case. Protected literals.

- T. Straightforward.

- BA, where A is an atom.

  BA is true in $(I, \omega(\Sigma), S)$
  iff $\forall J \in \omega(\Sigma) : A$ is true in $(J, \omega(\Sigma), S)$
  iff $\forall J \in \omega(\Sigma) : A \in J$
  iff {set theory}
  $A \in \bigcap_{J \in \omega(\Sigma)} J$
  iff {Lemma 1(b)}
  $A \in \Sigma^p$
  iff $A \in \Sigma$
  iff $\Sigma \models \mathrm{B}A$

- B$\neg A$, where A is an atom.

  B$\neg A$ is true in $(I, \omega(\Sigma), S)$
  iff $\forall J \in \omega(\Sigma) : \neg A$ is true in $(J, \omega(\Sigma), S)$
  iff $\forall J \in \omega(\Sigma) : A$ is not true in $(J, \omega(\Sigma), S)$
  iff $\forall J \in \omega(\Sigma) : A \notin J$
  iff {set theory}
  $A \notin \bigcup_{J \in \omega(\Sigma)} J$
  iff {Lemma 1(c)}
  $A \in \Sigma^n$
  iff $\neg A \in \Sigma$
  iff $\Sigma \models \mathrm{B}\neg A$

(2) Induction step. The formula has the form $\neg F$ or $F \wedge G$, where F and G are PL-formulas. Straightforward. □

Let T be a PL-theory, and Σ a set of literals. We define an operator similar to Γ as below:

$$\Gamma_0(T, \Sigma) = \left\{ \Sigma' \subset Lit \;\middle|\; \begin{array}{l} \Sigma' \text{ is a minimal} \\ \text{closed set such that} \\ \forall F \in T : F \\ \text{is true in} \\ (I, \omega(\Sigma'), \omega(\Sigma)) \end{array} \right\}.$$

Note that I is an arbitrary interpretation; recall that, by Lemma 2, the choice of I has no effect on whether or not F is true relative to a triple (I, S, S').

Lemma 7. *Let I be an interpretation, and S a set of interpretations. Then (I, S) is a model of a PL-theory T if and only if S has the form $\omega(\Sigma)$ for a closed set of literals Σ such that $\Sigma \in \Gamma_0(T, \Sigma)$.*

Proof. We first define a predicate Φ as follows:

$$\Phi(X) \equiv \forall F \in T : F \text{ is true in } (I, X, S).$$

Clearly, if S_1 and S_2 are equivalent then $\Phi(S_1) \equiv \Phi(S_2)$.

Left to right: Let (I, S) be a model of T. Thus, $(I, S) \in \Gamma(T, S)$, which means that (I, S) is a maximal structure such that S satisfies Φ. That is:

(i) $\Phi(S)$ holds, and
(ii) if $S \subsetneq S'$, then $\Phi(S')$ does not hold.

By Lemma 3, there is a closed set Σ of literals such that $S \subset \omega(\Sigma)$ and S, $\omega(\Sigma)$ are equivalent.

We want to show:

(1) $S = \omega(\Sigma)$.
Suppose the contrary, that is, $S \subsetneq \omega(\Sigma)$. From the fact that S and $\omega(\Sigma)$ are equivalent and (i), we conclude that $\Phi(\omega(\Sigma))$ holds as well, which contradicts (ii).

(2) $\Sigma \in \Gamma_0(T, \Sigma)$.
To prove this, we need to show that Σ is a minimal closed set such that $\Phi(\omega(\Sigma))$ holds. From (1) and (i), $\omega(\Sigma)$ satisfies Φ. It remains to show that Σ is minimal. Assume that $\Sigma' \subsetneq \Sigma$. By (1) and Lemma 1(d), $S = \omega(\Sigma) \subsetneq \omega(\Sigma')$; by (ii), it follows that $\Phi(\omega(\Sigma'))$ does not hold.

Right to left: Let Σ be a set of literals such that $\Sigma \in \Gamma_0(T, \Sigma)$. This means that Σ is a minimal closed set such that $\omega(\Sigma)$ satisfies Φ. That is:

(i) $\Phi(\omega(\Sigma))$ holds and,
(ii) if $\Sigma' \subsetneq \Sigma$, then $\Phi(\omega(\Sigma'))$ does not hold.

To prove that $(I, \omega(\Sigma))$ is a model of T, we need to show that $(I, \omega(\Sigma)) \in \Gamma(T, \omega(\Sigma))$, which means that $(I, \omega(\Sigma))$ is a maximal structure such that $\omega(\Sigma)$ satisfies Φ. We know from (i) that $\omega(\Sigma)$ satisfies Φ. Assume that $\omega(\Sigma) \subsetneq S'$. By Lemma 3, there exists a closed set of literals Σ' such that $S' \subset \omega(\Sigma')$ and S' is equivalent to $\omega(\Sigma')$. Then $\omega(\Sigma) \subsetneq S' \subset \omega(\Sigma')$, and it follows by Lemma 1(d) that $\Sigma' \subsetneq \Sigma$. Then, by (ii), $\omega(\Sigma')$ does not satisfy Φ. Since this set is equivalent to S', we conclude that S' does not satisfy Φ either. □

Lemma 8. *Let T be a positive basic PL-theory, and Σ a set of literals. Then $\Gamma_0(T, \Sigma)$ is the set of all answer sets of T.*

Proof. Using Lemma 6, we can restate the definition of Γ_0 as

$$\Gamma_0(T, \Sigma) = \left\{ \Sigma' \subset Lit \;\middle|\; \begin{array}{l} \Sigma' \text{ is a minimal} \\ \text{closed set such that} \\ \forall F \in T : \Sigma' \models F \end{array} \right\}.$$

Clearly, the condition for being an element of $\Gamma_0(T, \Sigma)$ is exactly the same as that for an answer set of T. □

Lemma 9. *Let T be a basic PL-theory, and Σ a closed set of literals. Then*

$$\Gamma_0(T^\Sigma, \Sigma) = \Gamma_0(T, \Sigma).$$

Proof. By definition

$$\Gamma_0(T^{\Sigma}, \Sigma) = \left\{ \Sigma' \subset Lit \;\middle|\; \begin{array}{l} \Sigma' \text{ is a minimal} \\ \text{closed set such that} \\ \forall F \in T^{\Sigma}{:}F \\ \text{is true in} \\ (I, \omega(\Sigma'), \omega(\Sigma)) \end{array} \right\}.$$

Since T^{Σ} stands for $\{F^{\Sigma} : F \in T\}$, this can be rewritten as

$$\Gamma_0(T^{\Sigma}, \Sigma) = \left\{ \Sigma' \subset Lit \;\middle|\; \begin{array}{l} \Sigma' \text{ is a minimal} \\ \text{closed set such that} \\ \forall F \in T{:}F^{\Sigma} \\ \text{is true in} \\ (I, \omega(\Sigma'), \omega(\Sigma)) \end{array} \right\}.$$

Lemma 5 shows that the right-hand side is equal to $\Gamma_0(T, \Sigma)$. □

Theorem 1 (Part A). *A structure (I, S) is a model of a basic PL-theory T if and only if $S = \omega(\Sigma)$ for some answer set Σ of T.*

Proof. By Lemma 7, it is sufficient to show that, for any set of literals Σ, Σ is an answer set of T iff $\Sigma \in \Gamma_0(T, \Sigma)$. By the definition of answer sets for basic PL-theories, Σ is an answer set of T iff Σ is an answer set of T^{Σ}. By Lemma 8, this can be expressed as $\Sigma \in \Gamma_0(T^{\Sigma}, \Sigma)$. By Lemma 9, this is equivalent to $\Sigma \in \Gamma_0(T, \Sigma)$. □

Lemma 10. *Let I be an interpretation, S, S' sets of interpretations, and F a PL-formula. Then*

$$F^{\star} \text{ is true in } (I, S, S') \text{ iff } F \text{ is true in } (I, S, S').$$

Proof. By structural induction.

(1) Base case. Protected literals. $F^{\star}$ coincides with F.

(2) Induction step.

- $\neg F$. Straightforward.
- BF. Using Lemma 2, we can conclude that

$$F \text{ is true in } (I, S, S') \text{ iff } \mathsf{B}F \text{ is true in } (J, S, S') \tag{9}$$

for any PL-formula F, any nonempty S, and any S'.

($\mathsf{B}F$)* is true in (I, S, S')
iff $F^{\star} \vee \mathsf{B}F$ is true in (I, S, S')
iff $F^{\star}$ is true in (I, S, S') or
$\mathsf{B}F$ is true in (I, S, S')
iff {ind. hyp.}
F is true in (I, S, S') or
$\mathsf{B}F$ is true in (I, S, S')
iff F is true in (I, S, S') or $S = \emptyset$
iff {(9)}
BF is true in (I, S, S') or $S = \emptyset$
iff BF is true in (I, S, S')

- *not* F. Using Lemma 2, we can conclude that

$$\neg F \text{ is true in } (I, S, S') \text{ iff } \textit{not } F \text{ is true in } (J, S, S') \tag{10}$$

for any PL-formula F, any S, and any nonempty S'.

(*not* F)* is true in (I, S, S')
iff $\neg F^{\star} \wedge \mathsf{not}\ \mathsf{F}$ is true in (I, S, S')
iff $\neg F^{\star}$, $\mathsf{not}\ \mathsf{F}$ are true in (I, S, S')
iff {ind. hyp.}
$\neg F$, $\mathsf{not}\ \mathsf{F}$ are true in (I, S, S')
iff $\neg F$ is true in (I, S, S') and $S' \neq \emptyset$
iff {(10)}
not F is true in (I, S, S') and $S' \neq \emptyset$
iff *not* F is true in (I, S, S')

- $F \wedge G$. Straightforward.

□

Lemma 11. *Let I be an interpretation, S a set of interpretations, and T a PL-theory. Then (I, S) is a model of T if and only if (I, S) is a model of $T^{\star}$.*

Proof. Immediate from Lemma 10. □

Theorem 1 (Part B). *A structure (I, S) is a model of a PL-theory T if and only if $S = \omega(\Sigma)$ for some answer set Σ of T.*

Proof. By Lemma 11, $T^{\star}$ and T have the same models; by the definition of an answer set, they have the same answer sets. Consequently, the statement of the theorem follows from Theorem 1 (Part A). □

A3. Proofs of Theorem 2 and Theorem 3

Lemma 12. *Let F be a basic PL-formula, and X, Σ closed sets of literals. Let F' be the formula obtained from F by substituting* T *for all occurrences of* BT, *and* F *for all occurrences of not* T. *Then*

$$X \models F^{\Sigma} \text{ iff } X \models (F')^{\Sigma}.$$

Proof. It is clear that $(F')^{\Sigma}$ can be obtained from F^{Σ} by substituting T for all occurrences of BT. □

Theorem 2. *Let T be a basic PL-theory, and let T' be obtained from it by substituting T for all occurrences of BT, and F for all occurrences of not T, in every axiom. Then T and T' have the same answer sets.*

Proof. Clearly, $T' = \{F' \mid F \in T\}$. Let Σ be a closed set of literals.

Σ is an answer set of T'
iff Σ is an answer set of $(T')^{\Sigma}$
iff $\begin{cases} \forall F \in (T')^{\Sigma} : \Sigma \models F, \text{ and} \\ \text{for each } X \subsetneq \Sigma, \exists F \in (T')^{\Sigma} : X \not\models F \end{cases}$
iff $\begin{cases} \forall F \in T : \Sigma \models (F')^{\Sigma}, \text{ and} \\ \text{for each } X \subsetneq \Sigma, \exists F \in T : X \not\models (F')^{\Sigma} \end{cases}$
iff {Lemma 12}
$\begin{cases} \forall F \in T : \Sigma \models F^{\Sigma}, \text{ and} \\ \text{for each } X \subsetneq \Sigma, \exists F \in T : X \not\models F^{\Sigma} \end{cases}$
iff Σ is an answer set of T^{Σ}
iff Σ is an answer set of T

□

Lemma 13. *Let F be a basic PL-formula, and X, Σ closed sets of literals. Let F' be the formula obtained from F by substituting F for all occurrences of BF, and T for all occurrences of not F. Suppose $\Sigma \neq Lit$. Then*

$$X \models F^{\Sigma} \text{ iff } X \models (F')^{\Sigma}.$$

Proof. If $\Sigma \neq Lit$, then $\mathsf{F} \notin \Sigma$, and consequently $(F')^{\Sigma}$ can be obtained from F^{Σ} by substituting F for all occurrences of BF. □

Theorem 3. *Let T be a basic PL-theory, and let T' be obtained from it by substituting F for all occurrences of BF, and T for all occurrences of not F, in every axiom. For any set of literals Σ other than Lit, Σ is an answer set of T if and only if it is an answer set of T'.*

Proof. Similar to the proof of Theorem 2, except that only answer sets different from Lit are considered. □

A4. Proof of Theorem 4

Lemma 14. *Let F be a basic PL-formula that contains no positive occurrences of not, and let Σ, Σ' be two closed sets of literals such that $\Sigma \subsetneq \Sigma'$. Then, for any closed set of literals X, if $X \models F^{\Sigma}$ then $X \models F^{\Sigma'}$.*

Proof. Since F is a basic PL-formula, we can rewrite it in conjunctive normal form as

$$G_1 \wedge \ldots \wedge G_n,$$

where all occurrences of *not* in each G_i are preceded by a negation. Thus it follows easily that, for any closed set of literals X, if $X \models G_i^{\Sigma}$ then $X \models G_i^{\Sigma'}$. Hence if $X \models F^{\Sigma}$ then $X \models F^{\Sigma'}$. □

Theorem 4. *If the axioms of a PL-theory contain no positive occurrences of not, then it cannot have two answer sets of which one is a proper subset of the other.*

Proof. It is sufficient to prove the theorem for basic PL-theories, because *not* occurs positively in a general PL-theory if and only if it occurs positively in the basic PL-theory T^*. Let T be a basic PL-theory whose axioms contain no positive occurrences of *not*. Let Σ, Σ' be two answer sets of T such that $\Sigma \subsetneq \Sigma'$. As Σ is an answer set of T, we have, for all axioms $F \in T$, $\Sigma \models F^{\Sigma}$. By Lemma 14, this implies for all axioms $F \in T$, $\Sigma \models F^{\Sigma'}$. This contradicts the minimality of Σ' among all closed sets satisfying the axioms of $T^{\Sigma'}$. □

RS theory: a Really Skeptical theory of Inheritance with Exceptions

Geneviève Simonet
L.I.R.M.M. 860,rue St Priest
34090 Montpellier France
email : simonet@crim.fr

Abstract

This paper presents two results in path-based defeasible inheritance theory defined with off-path without reinstatement preemption. First a definition of an extension in this theory through a fixedpoint equation and a polynomial algorithm to compute it are given. Second RS theory, Really Skeptical theory, is defined, which is intended to make up for the deficiencies of previously proposed approaches.

1 INTRODUCTION

In Knowledge Representation systems, knowledge is often organized hierarchically in the form of simple or multiple inheritance graphs, with or without exceptions. In this paper, I consider the problem of defining and computing the conclusion set of a multiple inheritance graph Γ with exceptions. Different theories have been proposed to solve it: path-based theories, by D.S.Touretzky, J.F.Horty and R.H.Thomason ([Tou86], [Hor91a], [THT87], [HTT90], [TTH91], [Hor91b]), E.Sandewall [San86], B.Selman and H.J.Levesque [SL91], E.Gregoire [Gre90b], L.A.Stein ([Ste91], [Ste89]) and K.Schlechta ([Sch90], [MS91], [Sch92]) and methods based on nonmonotonic logics, by R.H.Thomason and J.F.Horty [TH89], E.Gregoire [Gre90a], D.L.Poole [Poo85] and G.Brewka [Bre87].

In this paper, I will study path-based defeasible credulous and skeptical inheritance theories, defined with construction by upward concatenation and off-path without reinstatement preemption. I will first show, as an answer to Touretzky et al. [TTH91], that an extension of Γ in such a theory can be defined through a fixedpoint equation and efficiently computed.

It has been originally pointed out by Touretzky et al. [THT87] that skeptical theory is not so skeptical as one could expect, as ambiguities are blocked instead of being propagated upwards. E.Gregoire [Gre90b] and L.A.Stein [Ste89] have tried to improve skeptical theory by ambiguity propagation. L.A. Stein and K.Schlechta define the ideal skeptical conclusion set as the intersection of credulous ones. K.Schlechta proves that computing this intersection is untractable, and impossible under some natural assumptions [Sch90] [Sch92]. L.A. Stein proposes a polynomial algorithm to compute it, but with a different notion of extension from the one used in this paper [Ste89]. I will define RS theory (*Really Skeptical* theory), verifying the following property:
A conclusion of Γ in this theory is a conclusion of any credulous extension of Γ,
and discuss the difficulty of computing the intersection of the credulous extensions or conclusion sets of Γ, thus completing K.Schlechta's results.

2 NOTATIONS AND BASIC DEFINITIONS

Let $\Gamma = (X, U)$ be a multiple inheritance graph, i.e. a finite directed acyclic graph.
The elements may represent classes, instances of classes, types, objects, etc... according to the application.
Let x and y be two vertices of Γ.

$x \longrightarrow y$ denotes a positive link in Γ from x to y (relation of type 'is-a').

$x \longrightarrow \bar{y}$ denotes a negative link in Γ from x to y (relation of type 'is-not-a').

$\alpha : x \xrightarrow{\alpha} y$ denotes a positive path in Γ from x to y (every link of α is positive)

(α may be reduced to a point).

$\alpha : x \xrightarrow{\alpha} \bar{y}$ denotes a negative path in Γ from x to y (the last link of α is its unique negative one).
In path-based theories, one draws from Γ a set of positive and negative paths, called extension of Γ and noted Φ, and the conclusion set C of Γ is defined as

the set of conclusions *supported* by Φ, i.e.:

$C = C(\Phi) = \{x \rightarrow y /$ there is a positive path in Φ from x to $y\} \cup \{x \rightarrow \bar{y} /$ there is a negative path in Φ from x to $y\}$

$C_\Gamma^{+[-]}(x,y)$ denotes the set of not reduced to a point positive [negative] paths in Γ from x to y.
(subscript 'Γ' will be omitted when there is no ambiguity)

$\Phi^{+[-]}(x,y)$ denotes the set of positive [negative] paths in Φ from x to y.

$V(x,y) = \{x \rightarrow y, x \rightarrow \bar{y}\} \cap U$

$C_\Gamma(x,y) = C_\Gamma^+(x,y) \cup C_\Gamma^-(x,y)$

$\Phi(x,y) = \Phi^+(x,y) \cup \Phi^-(x,y)$

$C_\Gamma(x)$ [$\Phi(x)$] denotes the set of not reduced to a point positive and negative paths in Γ [Φ] from x.

A formula with $x \rightarrow y$ (resp. $C^+(x,y)$, $\Phi^+(x,y)$) implicitly admits a dual formula with $x \rightarrow \bar{y}$ (resp. $C^-(x,y)$, $\Phi^-(x,y)$).
$C(\Phi)$ is thus defined by:
$x \rightarrow y \in C(\Phi)$ iff $\Phi^+(x,y) \neq \emptyset$

Φ is a set of not reduced to a point positive and negative paths in Γ defined as follows:

1. Credulous theory:
C-inheritability in Φ is defined by:
Let α be an element of $C^+(x,y)$.

case 1: α is a link
Then α is C-inheritable in Φ.

case 2: α is a compound path
$\alpha : x \stackrel{\alpha_1}{\rightarrow} z \rightarrow y$
We define:

- α is *constructible in* Φ iff $\alpha_1 \in \Phi$
  (construction by upward concatenation)
- α' *contradicts* α iff $\alpha' \in C^-(x,y)$
- α is *contradicted in* Φ iff there is α' in Φ such that α' contradicts α.
- α is *preempted in* Φ iff there are z', α'_1, α'_2 such that:
  1) $z' \neq z$
  2) $z' \rightarrow \bar{y} \in U$
  3) $x \stackrel{\alpha'_1}{\rightarrow} z' \stackrel{\alpha'_2}{\rightarrow} z \in \Phi$
  (off-path preemption)
- α is *C-inheritable in* Φ iff:
  C.1) α is constructible in Φ
  C.2) α is not contradicted in Φ
  C.3) α is not preempted in Φ.

Φ is a credulous extension of Γ iff for any vertices x,y of Γ and any path α of $C^+(x,y)$:
$\alpha \in \Phi$ iff α is C-inheritable in Φ.

2. Skeptical theory:
A skeptical extension is defined from S-inheritability as above, just replacing conditions C.1),C.2) and C.3) by S.1), S.2) and S.3):

S.1) α is constructible in Φ

S.2) α is not preempted in Φ

S.3) Any compound path constructible in Φ and contradicting α is preempted in Φ.[1]
It will be shown in section 3 that Γ has a unique skeptical extension.
Proofs will often go by induction on $deg_\Gamma(x,y)$, which denotes the maximum length of a path in Γ from x to y. By convention, if there is no path from x to y, $deg_\Gamma(x,y)$ is equal to -1.
<1: (resp. :1>) marks the beginning (resp. end) of the proof of Proposition 1.
<1.1: (resp. :1.1>) marks the beginning (resp. end) of the proof of lemma 1.1.

3 A POLYNOMIAL ALGORITHM TO COMPUTE AN EXTENSION

J.F. Horty, R.H.Thomason and D.S.Touretzky propose a polynomial algorithm to compute the unique skeptical with reinstatement extension of Γ, using parallel marker propagation [HTT90]. In [TTH91] [2], they consider that without reinstatement preemption is semantically more correct, but less efficiently computable than with reinstatement one. They think that it cannot be computed using parallel marker propagation with a bounded number of markers and they do not know any definition of a skeptical without reinstatement extension through a fixedpoint equation. I have given such a definition in section 2 and will now show that it can be efficiently computed using parallel marker propagation with a bounded number of markers, just marking links instead of vertices. (I will not explicitly develop this parallel algorithm, but it will be straightforward from Horty et al.'s algorithm [HTT90] and following results.)
We first show Proposition 1:

[1] Above definitions imply without reinstatement preemption. Reinstatement is studied by Touretzky et al. [TTH91] and G.Simonet [Sim91]. To get with reinstatement definitions, just replace conditions C.3) and S.2) by:
C.3) $x \rightarrow y \in U$ or there is a compound path of $C^+(x,y)$, constructible in Φ and not preempted in Φ.
S.2) $x \rightarrow \bar{y} \notin U$.
Horty et al. in [Hor91a] and [HTT90], Selman et al. [SL91] and E.Gregoire [Gre90b] implicitly define credulous (resp. skeptical) theory without (resp. with) reinstatement.

[2] Article [TTH91] is a short one referring to [Hor91b] for more details. In particular, the definition of a skeptical extension without reinstatement is not given in [TTH91]. Unfortunately, I cannot find article [Hor91b].

Proposition 1:
In any defeasible credulous or skeptical theory using construction by upward concatenation and (off-path or with reinstatement preemption), for any extension Φ and vertex x of Γ, there is a subgraph $\Gamma(x)$ of Γ such that:
$\Phi(x) = C_{\Gamma(x)}(x)$

<1: We show that such a theory verifies property Q below (lemma 1.1), which is a sufficient condition of existence of $\Gamma(x)$ (lemma 1.2).
Property Q:

If $\alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in \Phi$ and $\alpha' : x \xrightarrow{\alpha'_1} z \rightarrow y, \alpha'_1 \in \Phi$,

then ($\alpha \in \Phi$ iff $\alpha' \in \Phi$).
(Note that theories using on-path **and** without reinstatement preemption verify neither property Q nor Proposition 1.)

Lemma 1.1:
Any defeasible credulous or skeptical theory using construction by upward concatenation and (off-path **or** with reinstatement preemption) verifies property Q.
<1.1: We suppose $\alpha \in \Phi$. Let us show that$\alpha' \in \Phi$.
α' verifies C.1) and S.1) because $\alpha'_1 \in \Phi$.
α' verifies C.2) and S.3) because α verifies them.
α' verifies C.3) and S.2 because α verifies them and preemption is off-path **or** with reinstatement.
So $\alpha' \in \Phi$.
We have in the same way: if $\alpha' \in \Phi$ then $\alpha \in \Phi$. :1.1>

Lemma 1.2:
In any defeasible theory verifying property Q and using construction by upward concatenation, for any extension Φ and vertex x of Γ, there is a subgraph $\Gamma(x)$ of Γ such that:

$\Phi(x) = C_{\Gamma(x)}(x)$
<1.2: We note:

$\Gamma(x) = (X, U(x))$, $U(x) = \cup\, U(x,y), y \in X$,
where $U(x,y)$ is the set of links of $U(x)$ into y.
We define $U(x,y)$ by:

$U(x,y) = \{z \rightarrow y[\bar{y}] / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y[\bar{y}], \alpha \in \Phi\}$
Let us show by induction on $deg_\Gamma(x,y)$ that for any y in X, $\Phi(x,y) = C_{\Gamma(x)}(x,y)$
If $deg_\Gamma(x,y) \leq 0$ then it is trivially true.
We suppose it is true for $deg_\Gamma(x,y) \leq k$ ($k \geq 0$). Let us show that it is true for $deg_\Gamma(x,y) = k+1$.
$\Phi(x,y) = V(x,y) \cup \{\alpha : x \xrightarrow{\alpha_1} z \rightarrow y[\bar{y}], \alpha_1 \in \Phi^+(x,z), z \rightarrow y[\bar{y}] \in U(x,y)\}$ (from property Q and construction by upward concatenation)
$\Phi(x,y) = V(x,y) \cup \{\alpha : x \xrightarrow{\alpha_1} z \rightarrow y[\bar{y}], \alpha_1 \in C^+_{\Gamma(x)}(x,z), z \rightarrow y[\bar{y}] \in U(x,y)\}$ (induction)
$\Phi(x,y) = C_{\Gamma(x)}(x,y)$:1.2> :1>

In this paper, I will study off-path without reinstatement preemption. Analogous results are obtained in the case of off-path with reinstatement preemption, using either link-marking (as in this paper) or vertex-marking (as Horty et al. did in [HTT90]).
Let us show that an extension of Γ can be efficiently computed.
We note:

$ST(x,y) = \{z \in X / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in \Phi\}$

$SF(x,y) = \{z \in X / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow \bar{y}, \alpha_1 \in \Phi\}$

$ST_1(x,y) = \{z \in ST / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in \Phi$ and α is not preempted in $\Phi\}$

$SF_1(x,y) = \{z \in SF / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow \bar{y}, \alpha_1 \in \Phi$ and α is not preempted in $\Phi\}$
In the following, ST (resp. SF, ST_1, SF_1) will implicitly denote $ST(x,y)$ (resp. ... , $SF_1(x,y)$).

We have:
If $V(x,y) = \{x \rightarrow y, x \rightarrow \bar{y}\}$ then any compound path constructible in Φ is preempted in Φ by the link contradicting it, $ST_1 = SF_1 = \emptyset$, and $U(x,y) = \{x \rightarrow y, x \rightarrow \bar{y}\}$.
If $V(x,y) = \{x \rightarrow y\}$ then $SF_1 = \emptyset$ and $U(x,y) = \{x \rightarrow y\} \cup \{z \rightarrow y, z \in ST_1\}$.
If $V(x,y) = \{x \rightarrow \bar{y}\}$ then $ST_1 = \emptyset$ and $U(x,y) = \{x \rightarrow \bar{y}\} \cup \{z \rightarrow \bar{y}, z \in SF_1\}$.
If $V(x,y) = \emptyset$ then:
If $ST_1 = \emptyset$ and $SF_1 = \emptyset$ then $U(x,y) = \emptyset$,
If $ST_1 \neq \emptyset$ and $SF_1 = \emptyset$ then $U(x,y) = \{z \rightarrow y, z \in ST_1\}$,
If $ST_1 = \emptyset$ and $SF_1 \neq \emptyset$ then $U(x,y) = \{z \rightarrow \bar{y}, z \in SF_1\}$,
If $ST_1 \neq \emptyset$ and $SF_1 \neq \emptyset$ then:

In credulous theory:
Two values are possible:
$U(x,y) = \{z \rightarrow y, z \in ST_1\}$ or $U(x,y) = \{z \rightarrow \bar{y}, z \in SF_1\}$

In skeptical theory:
$U(x,y) = \emptyset$.

We immediately obtain the following recursive definition of $U(x,y)$:

Function $U(x,y)$:

If $ST_1 = \emptyset$ or $SF_1 = \emptyset$ then $U(x,y) = V(x,y) \cup \{z \rightarrow y, z \in ST_1\} \cup \{z \rightarrow \bar{y}, z \in SF_1\}$

else $\{V(x,y) = \emptyset\}$ $\{conflict\}$

In credulous theory:
Two values are possible:
$U(x,y) = \{z \rightarrow y, z \in ST_1\}$ or $U(x,y) = \{z \rightarrow \bar{y}, z \in SF_1\}$

In skeptical theory:
$U(x,y) = \emptyset$

The recursivity comes from the computation of ST_1

and SF_1 from the equivalence:
$\forall \alpha : x \xrightarrow{\alpha_1} z \rightarrow y$,
$\alpha_1 \in \Phi$ iff $\forall u : z' \rightarrow y'$ link of $\alpha_1, u \in U(x, y')$
Termination of the recursivity is insured by the fact that:
$\forall u : z' \rightarrow y'$ link of $\alpha_1, deg_\Gamma(x, y') < deg_\Gamma(x, y)$

We can see from this algorithm that there is a unique skeptical extension of Γ, and at least one credulous one, with more than one iff Γ contains at least one conflict.

Complexity:

Leaving aside the computation of ST_1 and SF_1, computing $U(x, y)$ is polynomial. Can ST_1 and SF_1, be polynomially computed? One shows that the sets ST, SF, ST_1 and SF_1 may be rewritten as follows:

$ST = \{z \in X / z \rightarrow y \in U$ and $U(x, z)$ contains a positive link$\}$

$SF = \{z \in X / z \rightarrow \bar{y} \in U$ and $U(x, z)$ contains a positive link$\}$

$ST_1 = \{z \in ST / x \rightarrow \bar{y} \notin U$ and $SF \cap D(x, z) = \emptyset\}$

$SF_1 = \{z \in SF / x \rightarrow y \notin U$ and $ST \cap D(x, z) = \emptyset\}$
where $D(x, z) = \{w \in X / C^+_{\Gamma(x)}(w, z) \neq \emptyset\}$
We may rewrite function $U(x, y)$ as follows:

Function $U(x, y)$:

For every z s.t. $\exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y[\bar{y}]$ and $U(x, z)$ has not been computed yet do compute $U(x, z)$

Computation of ST and SF from the sets $U(x, z)$

{ At this step, for any w in X s.t. there is a positive path in Γ from x to w and a path in Γ from w to y, $U(x, w)$ has been computed, so that for any z in $ST \cup SF$, $D(x, z)$ is directly computable. }

Computation of ST_1 and SF_1 from ST, SF and the sets $D(x, z)$

Computation of $U(x, y)$ from the sets ST_1 and SF_1

In that form, computing $U(x, y)$ is clearly polynomial. Therefore, computing a credulous or the skeptical extension or conclusion set of Γ is of polynomial complexity.

4 IS SKEPTICAL THEORY REALLY SKEPTICAL?

The answer is 'no', as Touretzky et al. originally pointed it out [THT87]:
In figure 1,where $\Gamma(x)$ corresponds to the skeptical extension of Γ, one draws the conclusion that an x is not an y, whereas an x might be a z and an y. This situation is inadmissible if one wants the theory to adopt a really skeptical attitude. Suppose, for instance, that the vertices in figure 1 represent classes of mushrooms, and y the class of poisonous ones. Imagine what may be the consequences of concluding that a mushroom of class x is not poisonous whereas it is ! The ambiguity between x and z is blocked at vertex z, whereas it should be propagated upwards and introduce a conflict between x and y.

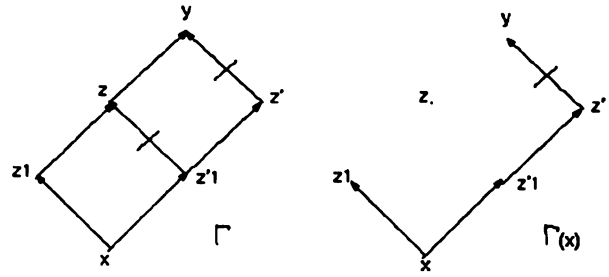


Figure 1: Is Skeptical Theory Really Skeptical?

E.Gregoire and L.A.Stein have tried to improve skeptical theory through ambiguity propagation. E.Gregoire, in [Gre90b], makes it more expressive, but not more skeptical, as the conclusion set is unchanged. In [Ste89], L.A.Stein proposes ***ambiguity propagating inheritance*** and improves it into ***ideally skeptical inheritance*** (see section 5) to make up for the deficiencies of ***ambiguity blocking inheritance*** , but her definitions of extension and preemption are quite different from those used here. She uses a variant of on-path preemption, and I think, as it comes out of a discussion of Touretzky et al. [THT87], that off-path preemption should be preferred to on-path one.

5 WHAT SHOULD BE THE IDEAL SKEPTICAL THEORY?

The anomaly in figure 1 can be expressed in other words as:

$x \rightarrow y$ is a conclusion of the skeptical extension of Γ whereas there is a credulous extension of which it is not.
or

$\alpha : x \rightarrow z'_1 \rightarrow z' \rightarrow y$ is a path in the skeptical extension of Γ whereas there is a credulous extension in which it is not.
So we implicitly consider that the ideal skeptical theory should verify:
PIC: The skeptical conclusion set of Γ is the intersection of the credulous ones,
or
PIE: The skeptical extension of Γ is the intersection of the credulous ones,
the set of credulous extensions of Γ representing the set of 'possible cases'.

Both formulations are found in the literature: PIE one by Horty et al. [HTT90] and L.A.Stein (*ambiguity propagating inheritance* [Ste89]) and PIC one by L.A.Stein in [Ste89], where a polynomial algorithm is given to compute the *ideally skeptical* conclusion set verifying PIC, but, as already mentioned, with quite different definitions of extension and preemption from those used here. PIC property should be preferred to PIE one, because, as discussed by L.A.Stein [Ste89]and D.Makinson and K.Schlechta [MS91], a conclusion drawn in every credulous extension of Γ, but supported by different paths ('floating' conclusion in [MS91]), should be drawn in the skeptical extension of Γ (supported by which path? A partial answer will be given to this question in Proposition 5.)

Another question could be: does the set of credulous extensions cover all possible cases?
For instance, in figure 1, in every credulous extension of Γ, either conclusion $x \rightarrow z$ or conclusion $x \rightarrow \bar{z}$ is drawn. Should not we consider a third possibility: an x is neither a z, nor not a z, so that no conclusion is drawn between x and z?
We are thus led to define a 3-credulous extension of Γ, where function $U_3(x,y)$ is obtained from function $U(x,y)$ defined in section 3 by considering 3 possible values of $U_3(x,y)$ in case of conflict: $\{z \rightarrow y, z \in ST_1\}$, $\{z \rightarrow \bar{y}, z \in SF_1\}$ and $\emptyset$. A 3-credulous extension can be defined as follows:

Φ is a 3-credulous extension of Γ iff there is a subset Z of X^2 such that

$\forall (x,y) \in Z$ (resp. $X^2 - Z$), $\forall \alpha \in C^+(x,y)$,

$\alpha \in \Phi$ iff α is C-inheritable (resp. S-inheritable) in Φ.
The set of possible cases is covered by the set of 3-credulous extensions rather than credulous ones. But it follows from proposition 2 below that we do not lose anything if we only take credulous conclusion sets (resp. extensions) of Γ into account.
One shows:

Proposition 2:
The intersection of the 3-credulous conclusion sets (resp. extensions) of Γ is equal to the intersection of the credulous ones.

(Note that we would still have the equality, replacing 3-credulous by 4-credulous, in case we wished to consider a fourth possibility in case of conflict between x and z: an x is both a z and not a z.)

We could of course obtain a skeptical conclusion set or extension verifying PIC or PIE by computing all the credulous extensions of Γ, but we exclude this possibility, as we want tractability and the number of credulous extensions may be exponential in the size of Γ.
In the following:

Φ denotes the skeptical extension of Γ.

Φ' denotes a credulous extension of Γ.

C (resp. $C(\Phi')$) denotes the conclusion set of Φ (resp. Φ').

ST_1 (resp. $ST_1(\Phi')$) denotes the set ST_1 associated to Φ (resp. Φ') (vertices x and y being given).
SF_1 (resp. $SF_1(\Phi')$) denotes the set SF_1 associated to Φ (resp. Φ') (vertices x and y being given).

$U(x,y)$ (resp. $U_{\Phi'}(x,y)$) denotes the function $U(x,y)$ associated with Φ (resp. Φ').

$I\Phi$ denotes the intersection of the credulous extension of Γ.
$U\Phi$ denotes the union of the credulous extension of Γ.
[3]

$\forall X \in \{C, ST_1, SF_1\}$,
IX denotes the intersection of the sets $X(\Phi')$, Φ' credulous.
UX denotes the union of the sets $X(\Phi')$, Φ' credulous.

C_{SK} denotes $UC - IC$.

Properties PIC and PIE can be rewritten as:
PIC: $C = IC$
PIE: $\Phi = I\Phi$

We show that:

Proposition 3:
If Φ is the skeptical extension of Γ, then:
$C \supseteq IC$ and $\Phi \supset I\Phi$

<3:

$$C \supseteq \bigcap_{\Phi' \ 3-cred.} C(\Phi') = IC \text{ and } \Phi \supseteq \bigcap_{\Phi' \ 3-cred.} \Phi' = I\Phi,$$

where the inclusions follow from the fact that Φ is a particular 3-credulous extension of Γ and the equalities from Proposition 2. :3>
Figure 1 provides a counterexample of the reverse inclusion.

The two following propositions show the difficulty of defining a skeptical theory verifying PIC or PIE.

Proposition 4:
Computing the intersection (or union) of the credulous conclusion sets or extensions of Γ is NP-hard.

<4: we have to prove the NP-hardness of IC (resp. UC, $I\Phi$, $U\Phi$)-computing problem. That of IC-computing problem has been proved by K.Schlechta from a simple reduction from SAT-problem [Sch92]. That of UC-computing problem follows from the same reduction. Those of $I\Phi$ and $U\Phi$-computing problems can be proved from the following reduction from $IC(x,y)$ and $UC(x,y)$-computing ones, whose NP-hardness immediately follows from that of IC and UC-computing ones.
Let $\Gamma = (X,U)$ be an inheritance graph and x, y two

[3] A path of $U\Phi - I\Phi$ is called a 'zombie' path by D.Makinson and K.Schlechta in [MS91].

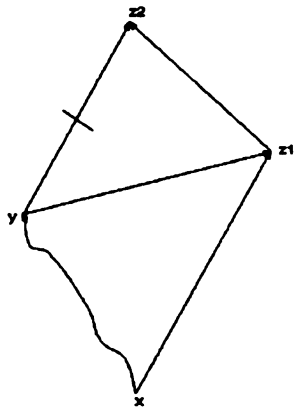


Figure 2: (Proof of Proposition 4)

distinct vertices of Γ. Let $\Gamma' = (X \cup \{z_1, z_2\}, U \cup \{x \rightarrow z_1, y \rightarrow z_1, y \rightarrow z_2, z_1 \rightarrow z_2\})$, where z_1 and z_2 are not in X, and let $\alpha : x \rightarrow z_1 \rightarrow z_2$ (see figure 2). We have:

$x \rightarrow y \in IC$ in Γ iff $\alpha \notin U\Phi$ in Γ'

$x \rightarrow y \in UC$ in Γ iff $\alpha \notin I\Phi$ in Γ' :4>

Proposition 5:
If the definition of Φ verifies:
a) $x \rightarrow y \in C$ iff $\Phi^+(x, y) \neq \emptyset$
b) If α is a compound path and $\alpha \in \Phi$, then α is constructible in Φ,
(resp. verifies:
c) If $\alpha \in \Phi$ then any compound path constructible in Φ and contradicting α is preempted in Φ,)
then property PIC (resp. PIE) is not always true.

<5: Suppose conditions a), b) and property PIC are always true.
In figure 3, $x_1 \rightarrow y_1 \in IC$, $x_1 \rightarrow z_2 \notin IC$, $x_1 \rightarrow z_3 \notin IC$.
We have:
$x_1 \rightarrow y_1 \in C$ (property PIC)
$\Phi^+(x_1, y_1) \neq \emptyset$ (condition a))
$\Phi^+(x_1, z_2) \neq \emptyset$ or $\Phi^+(x_1, z_3) \neq \emptyset$ (condition b))
$x_1 \rightarrow z_2 \in C$ or $x_1 \rightarrow z_3 \in C$ (condition a))
$x_1 \rightarrow z_2 \in IC$ or $x_1 \rightarrow z_3 \in IC$ (property PIC)
Contradiction.

Suppose condition c) and property PIE are always true.
In figure 3, $\alpha : x \rightarrow z \rightarrow y \in I\Phi$ and $\forall \mu : x \xrightarrow{\mu_1} z \xrightarrow{\mu_2} z', \mu \notin I\Phi$.
We have:
$\alpha \in \Phi$ (property PIE)
$\alpha' : x \rightarrow z' \rightarrow \bar{y}$ is preempted in Φ (condition c))
$\exists \mu : x \xrightarrow{\mu_1} z \xrightarrow{\mu_2} z', \mu \in \Phi$
$\exists \mu : x \xrightarrow{\mu_1} z \xrightarrow{\mu_2} z', \mu \in I\Phi$ (property PIE)
Contradiction. :5>

Concerning property PIC, K.Schlechta proves a somewhat technical stronger result: assuming some more complex, but less restrictive prerequisites than conditions a) and b), the intersection of the credulous extensions of Γ cannot be captured [Sch90] [Sch92].

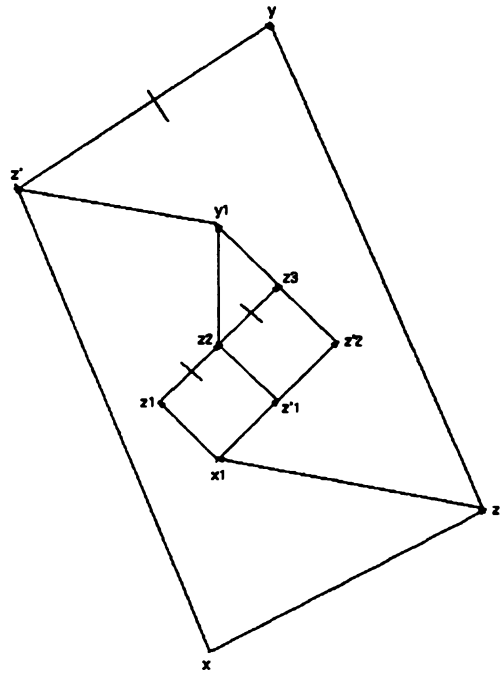


Figure 3: (Proof of Proposition 5)

As we will only consider theories verifying conditions a), b) and c) and in which C and Φ are polynomially computable, it follows from Proposition 4 or 5 that properties PIC and PIE will not always be true. We will define a *really skeptical* theory, i.e. only drawing conclusions that are drawn in every credulous extension (in other words, verifying the inclusion $C \subseteq IC$), trying to approach property PIC as closely as possible.

6 RS THEORY

6.1 DEFINITION

It follows from Proposition 1 that:

- $I\Phi(x) = C_{I\Gamma(x)}(x)$,

where $I\Gamma(x) = (X, IU(x))$, $IU(x) = \cup\, IU(x, y), y \in X$, $IU(x, y) = \cap\, U_{\Phi'}(x, y), \Phi'$*credulous*

- $U\Phi(x) \subseteq C_{U\Gamma(x)}(x)$,

where $U\Gamma(x) = (X, UU(x))$, $UU(x) = \cup\, UU(x, y), y \in X$, $UU(x, y) = \cup\, U_{\Phi'}(x, y), \Phi'$*credulous*
We show Proposition 6 below, where $I\Phi(\alpha)$ is defined as follows:

$\forall \alpha : x \xrightarrow{\alpha_1} z \rightarrow y$,

$I\Phi(\alpha) = \{\mu_1 \in C_\Gamma^+(x,z) / \forall u : z' \rightarrow y' \text{ link of } \mu_1, u \text{ is a link of } \alpha_1 \text{ or } u \in IU(x,y')\}$

Proposition 6:
$z \rightarrow y \in IU(x,y)$ iff $((z = x \text{ and } x \rightarrow y \in U)$ or $(USF_1 = \emptyset$ and $z \in IST_1))$
$z \rightarrow y \in UU(x,y)$ iff $((z = x$ and $x \rightarrow y \in U)$ or $z \in UST_1)$
$IST_1 \supseteq \{z \in X / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in I\Phi$ and α is not preempted in $U\Phi\}$
$UST_1 \subseteq \{z \in X / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in U\Phi$ and α is not preempted in $I\Phi(\alpha)\}$

(The reverse inclusions are not always true.)

<6: If $z = x$ then the two equivalences are trivially true.
If $z \neq x$ then we have:

1) $z \rightarrow y \in IU(x,y)$ iff $\forall \Phi'$ credulous, $z \rightarrow y \in U_{\Phi'}(x,y)$
$z \rightarrow y \in IU(x,y)$ iff $\forall \Phi'$ credulous, $z \in ST_1(\Phi')$ and $SF_1(\Phi') = \emptyset$
(If $SF_1(\Phi')$ was not empty, there would be a credulous extension Φ'' s.t. $z \rightarrow y \notin U_{\Phi''}(x,y)$)
$z \rightarrow y \in IU(x,y)$ iff $(USF_1 = \emptyset$ and $z \in IST_1)$

2) $z \rightarrow y \in UU(x,y)$ iff $\exists \Phi'$ credulous s.t. $z \rightarrow y \in U_{\Phi'}(x,y)$
$z \rightarrow y \in UU(x,y)$ iff $\exists \Phi'$ credulous s.t. $z \in ST_1(\Phi')$
$z \rightarrow y \in UU(x,y)$ iff $z \in UST_1$

3) Let $\alpha : x \xrightarrow{\alpha_1} z \rightarrow y$ s.t. $\alpha_1 \in I\Phi$ and α is not preempted in $U\Phi$. Let Φ' be a credulous extension. Let us show that $z \in ST_1(\Phi')$.
$\alpha_1 \in \Phi'$ and α is not preempted in Φ' (because $I\Phi \subseteq \Phi'$ and $U\Phi \supseteq \Phi'$)
$z \in ST_1(\Phi')$

4) Suppose $z \in UST_1$. Let us show that there is $\alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in U\Phi$ and α is not preempted in $I\Phi(\alpha)$.
$\exists \Phi'$ credulous s.t. $z \in ST_1(\Phi')$
Let $\alpha : x \xrightarrow{\alpha_1} z \rightarrow y$ s.t. $\alpha_1 \in \Phi'$ and α is not preempted in Φ'
$\alpha_1 \in U\Phi$ (because $\Phi' \subseteq U\Phi$)
To show that α is not preempted in $I\Phi(\alpha)$ it is sufficient to show that $\Phi' \supseteq I\Phi(\alpha)$.

Suppose $\mu_1 \in I\Phi(\alpha)$. Let us show that $\mu_1 \in \Phi'$.
$\forall u : z' \rightarrow y'$ link of μ_1, u is a link of α_1 or $u \in IU(x,y')$
$\forall u : z' \rightarrow y'$ link of $\mu_1, u \in U_{\Phi'}(x,y')$ (because $\alpha_1 \in \Phi'$ and $IU(x,y') \subseteq U_{\Phi'}(x,y')$)
$\mu_1 \in \Phi'$

So $\Phi' \supseteq I\Phi(\alpha)$, and therefore α is not preempted in $I\Phi(\alpha)$:6>

Note that $I\Phi$ can be defined through a fixedpoint equation. Defining *I-inheritability* (resp. *U-inheritability*) in Φ as C-inheritability in Φ, replacing conditions C.2) and C.3) by I.2) and I.3) (resp. U.2)):

I.2) $USF_1 = \emptyset$ U.2) $z \in UST_1$

I.3) $z \in IST_1$
we have:

$\alpha \in I\Phi$ iff α is I-inheritable in $I\Phi$.

If $\alpha \in U\Phi$ then α is U-inheritable in $U\Phi$.

We are thus led to define (or redefine already defined sets Φ, $\Gamma(x)$ and C) the sets:
Φ, $RU\Phi$, $\Gamma(x)$, $RU\Gamma(x)$, C, RUC, RC_{SK}, $RIST_1$, $RISF_1$, $RUST_1$ and $RUSF_1$
as approximations of:
$I\Phi$, $U\Phi$, $I\Gamma(x)$, $U\Gamma(x)$, IC, UC, C_{SK}, IST_1, ISF_1, UST_1 and USF_1 respectively as follows:

- Φ is the RS (*Really Skeptical*) extension of Γ (the adverb 'really' will be justified later) defined as the skeptical one, replacing S-inheritability by *RS-inheritability*, where conditions S.2) and S.3) are replaced by RS.2) and RS.3):
  RS.2) $RUSF_1 = \emptyset$
  RS.3) $z \in RIST_1$
- $RU\Phi$ is the set of 'possible' paths [4] defined by:
  $\alpha \in RU\Phi$ iff α is *RU-inheritable* in $RU\Phi$.
  where RU-inheritability in Φ is defined as C-inheritability in Φ, replacing conditions C.2) and C.3) by RU.2):
  RU.2) $z \in RUST_1$
- $\Gamma(x) = (X, U(x))$, $U(x) = \cup\, U(x,y), y \in X$, $U(x,y)$ is defined by:
  $z \rightarrow y \in U(x,y)$ iff $((z = x$ and $x \rightarrow y \in U)$ or $(RUSF_1 = \emptyset$ and $z \in RIST_1))$
  $RU\Gamma(x) = (X, RUU(x))$,
  $RUU(x) = \cup\, RUU(x,y), y \in X$, $RUU(x,y)$ is defined by:
  $z \rightarrow y \in RUU(x,y)$ iff $((z = x$ and $x \rightarrow y \in U)$ or $z \in RUST_1)$
- $C = C(\Phi)$, $RUC = C(RU\Phi)$, $RC_{SK} = RUC - C$
- $RIST_1 = \{z \in X / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in \Phi$ and α is not preempted in $RU\Phi\}$
  $RISF_1 = \{z \in X / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow \bar{y}, \alpha_1 \in \Phi$ and α is not preempted in $RU\Phi\}$
- $RUST_1 = \{z \in X / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in RU\Phi$ and α is not preempted in $\Phi(\alpha)\}$
  $RUSF_1 = \{z \in X / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow \bar{y}, \alpha_1 \in RU\Phi$ and α is not preempted in $\Phi(\alpha)\}$,
  where $\Phi(\alpha)$ is defined as $I\Phi(\alpha)$, replacing $IU(x,y')$ by '$U(x,y')$.

We have:

$\Phi(x) = C_{\Gamma(x)}(x)$

[4] A 'possible' path may be compared to a 'not totally discarded' path differently defined by E.Gregoire in [Gre90b].

$RU\Phi(x) = C_{RU\Gamma(x)}(x)$,
Therefore, the definitions of $U(x,y)$ and $RUU(x,y)$ are recursive in the same way as $U(x,y)$ in skeptical or a credulous theory (see section 3).

6.2 COMPLEXITY

Can $RIST_1$, $RISF_1$, $RUST_1$ and $RUSF_1$ be polynomially computed? Let ST, SF, $RUST$ and $RUSF$ be the sets defined by:

$ST = \{z \in X / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in \Phi\}$

$SF = \{z \in X / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow \bar{y}, \alpha_1 \in \Phi\}$

$RUST = \{z \in X / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in RU\Phi\}$

$RUSF = \{z \in X / \exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow \bar{y}, \alpha_1 \in RU\Phi\}$
One shows that all these sets may be rewritten as follows:

$ST = \{z \in X / z \rightarrow y \in U$ and $U(x,z)$ contains a positive link$\}$

$RUST = \{z \in X / z \rightarrow y \in U$ and $RUU(x,z)$ contains a positive link$\}$

$RIST_1 = \{z \in ST / x \rightarrow \bar{y} \notin U$ and $RUSF \cap RUD(x,z) = \emptyset\}$
where $RUD(x,z) = \{w \in X / C^+_{RU\Gamma(x)}(w,z) \neq \emptyset\}$

$RUST_1 = \{z \in RUST / C_{RUST\Gamma_1(x,y)}(x,z) \neq \emptyset\}$
where $RUST\Gamma_1(x,y) = (X, RUSTU_1(x,y)$,
$RUSTU_1(x,y) = \{u : z' \rightarrow y' \in U / u \in RUU(x,y'), z' \rightarrow \bar{y} \notin U$ and $SF \cap D(x,y') = \emptyset\}$ and $D(x,y') = \{w \in X / C^+_{\Gamma(x)}(w,y') \neq \emptyset\}$.

(idem for SF, $RUSF$, $RISF_1$ and $RUSF_1$)
Function $U(x,y)$ may be rewritten in the form of a procedure returning two results, $U(x,y)$ and $RUU(x,y)$, as follows:

Procedure $U - RUU(x,y; var\ U(x,y), RUU(x,y))$:

For every z s.t. $\exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y[\bar{y}]$ and $U - RUU(x,z,U(x,z),RUU(x,z))$ has not been called yet do $U - RUU(x,z,U(x,z),RUU(x,z))$

Computation of ST, SF, $RUST$ and $RUSF$ from the sets $U(x,z)$ and $RUU(x,z)$

{ At this step, for any w in X s.t. there is a positive path in Γ from x to w and a path in Γ from w to y, $U - RUU(x,w,U(x,w),RUU(x,w))$ has been called, so that for any z in $ST \cup SF$ (resp. $RUST$, $RUSF$), $RUD(x,z)$ (resp. $C_{RUST\Gamma_1(x,y)}(x,z)$, $C_{RUSF\Gamma_1(x,y)}(x,z)$) is directly computable. }

Computation of $RIST_1$, $RISF_1$, $RUST_1$ and $RUSF_1$ from ST, SF, $RUST$, $RUSF$ and the sets $RUD(x,z)$, $C_{RUST\Gamma_1(x,y)}(x,z)$ and $C_{RUSF\Gamma_1(x,y)}(x,z)$

Computation of $U(x,y)$ and $RUU(x,y)$ from the sets $RIST_1$, $RISF_1$, $RUST_1$ and $RUSF_1$

In that form, computing $U(x,y)$ is clearly polynomial. Therefore, computing the RS extension or conclusion set of Γ is of polynomial complexity.

6.3 COMPARISON WITH CREDULOUS AND SKEPTICAL THEORIES

Following proposition 7 shows that RS theory is *more skeptical* than credulous and skeptical ones.

Proposition 7:
If Φ be the RS extension and Φ' the skeptical or any credulous extension of Γ, then we have:
$\forall (x,y) \in X^2$,
$U(x,y) \subseteq IU(x,y) \subseteq U_{\Phi'}(x,y) \subseteq UU(x,y) \subseteq RUU(x,y)$
$\Phi(x,y) \subseteq I\Phi(x,y) \subseteq \Phi'(x,y) \subseteq U\Phi(x,y) \subseteq C_{U\Gamma(x)}(x,y) \subseteq RU\Phi(x,y)$
$RIST_1 \subseteq IST_1 \subseteq ST_1(\Phi') \subseteq UST_1 \subseteq RUST_1$
$RISF_1 \subseteq ISF_1 \subseteq SF_1(\Phi') \subseteq USF_1 \subseteq RUSF_1$

<7: Let us prove the inclusions:

$U(x,y) \subseteq U_{\Phi'}(x,y) \subseteq RUU(x,y)$

$RIST_1 \subseteq ST_1(\Phi') \subseteq RUST_1$

$RISF_1 \subseteq SF_1(\Phi') \subseteq RUSF_1$
(The other inclusions are trivial consequences of these ones.)
The proof goes by induction on $deg_\Gamma(x,y)$.
If $deg_\Gamma(x,y) \leq 0$ then the inclusions are trivially true.
We suppose they are true for $deg_\Gamma(x,y) \leq k$ $(k \geq 0)$.
Let us show that they are true for $deg_\Gamma(x,y) = k+1$.

1) Suppose $z \in RIST_1$. Let us show that $z \in ST_1(\Phi')$.
Let $\alpha : x \xrightarrow{\alpha_1} z \rightarrow y$ s.t. $\alpha_1 \in \Phi(x,z)$ and α is not preempted in $RU\Phi(x,z)$.
$\alpha_1 \in \Phi'(x,z)$ and α is not preempted in $\Phi'(x,z)$ (because $\Phi(x,z) \subseteq \Phi'(x,z)$ and $RU\Phi(x,z) \supseteq \Phi'(x,z)$ by induction hypothesis)
$z \in ST_1(\Phi')$

2) Suppose $z \in ST_1(\Phi')$. Let us show that $z \in RUST_1$.
Let $\alpha : x \xrightarrow{\alpha_1} z \rightarrow y$ s.t. $\alpha_1 \in \Phi'(x,z)$ and α is not preempted in $\Phi'(x,z)$
$\alpha_1 \in RU\Phi(x,z)$ (because $\Phi'(x,z) \subseteq RU\Phi(x,z)$ by induction)
To show that α is not preempted in $\Phi(\alpha)$ it is sufficient to show that $\Phi'(x,z) \supseteq \Phi(\alpha)$.

Suppose $\mu_1 \in \Phi(\alpha)$. Let us show that $\mu_1 \in \Phi'(x,z)$.
$\forall u : z' \rightarrow y'$ link of μ_1, u is a link of α_1 or $u \in U(x,y')$
$\forall u : z' \rightarrow y'$ link of μ_1, $u \in U_{\Phi'}(x,y')$ (because $\alpha_1 \in \Phi'$ and $U(x,y') \subseteq U_{\Phi'}(x,y')$ by induction)
$\mu_1 \in \Phi'(x,z)$

So $\Phi'(x,z) \supseteq \Phi(\alpha)$, and therefore α is not preempted in $\Phi(\alpha)$

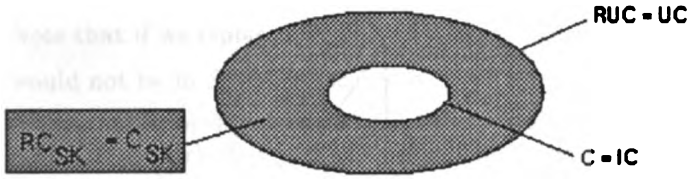


Figure 4: What we would Like to Have

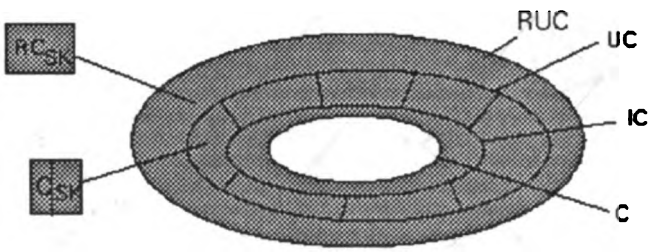


Figure 5: What we Have

$z \in RUST_1$
We prove in the same way that $RISF_1 \subseteq SF_1(\Phi') \subseteq RUSF_1$

3) Suppose $z \rightarrow y \in U(x,y)$. Let us show that $z \rightarrow y \in U_{\Phi'}(x,y)$.
If $z = x$ then $z \rightarrow y \in U_{\Phi'}(x,y)$.
If $z \neq x$ then we have:

$RUSF_1 = \emptyset$ and $z \in RIST_1$
$SF_1(\Phi') = \emptyset$ and $z \in ST_1(\Phi')$ (from the inclusions proved in 1) and 2))
$z \rightarrow y \in U_{\Phi'}(x,y)$

4) Suppose $z \rightarrow y \in U_{\Phi'}(x,y)$. Let us show that $z \rightarrow y \in RUU(x,y)$.
If $z = x$ then $z \rightarrow y \in RUU(x,y)$.
If $z \neq x$ then we have:

$z \in ST_1(\Phi')$
$z \in RUST_1$ (from the inclusion proved in 2))
$z \rightarrow y \in RUU(x,y)$:7>

Note that if $z \in ST$, then ($z \in RUST_1$ iff $z \in ST_1$). Therefore, as preemption in Φ is more simple than preemption in $\Phi(\alpha)$, in order to spare computing time, $RUST_1$ should be computed as the union of ST_1 and $SKST_1$, where:

$ST_1 = \{z \in ST / \exists \alpha : x \stackrel{\alpha_1}{\rightarrow} z \rightarrow y, \alpha_1 \in \Phi$ and α is not preempted in $\Phi\}$

$SKST_1 = \{z \in X - ST / \exists \alpha : x \stackrel{\alpha_1}{\rightarrow} z \rightarrow y, \alpha_1 \in RU\Phi$ and α is not preempted in $\Phi(\alpha)\}$,
i.e.

$ST_1 = \{z \in ST / x \rightarrow \bar{y} \notin U$ and $SF \cap D(x,z) = \emptyset\}$

$SKST_1 = \{z \in RUST - ST / C_{RUST\Gamma_1(x,y)}(x,z) \neq \emptyset\}$
(idem for $RUSF_1$ as union of SF_1 and $SKSF_1$).

Concerning the conclusion sets of Φ and $RU\Phi$, it follows from Proposition 7 that:

$C = C(\Phi) \subseteq C(I\Phi) \subseteq IC$

$RUC = C(RU\Phi) \supseteq C(U\Phi) = UC$
(The reverse inclusions are not always true.) (see figures 4 and 5.)

We have therefore:

Proposition 8:
If Φ is the RS extension of Γ, then:
$C \subseteq IC$ and $\Phi \subset I\Phi$

6.4 HOW COULD WE IMPROVE RS THEORY?

Could we improve the approximations of IST_1, UST_1, $I\Phi$ and $U\Phi$?
We have the inclusions:
$RIST_1 \subseteq IST_1$, $RISF_1 \subseteq ISF_1$,
$UST_1 \subseteq RUST_1$, $USF_1 \subseteq RUSF_1$,
$\Phi \subseteq I\Phi$ and $U\Phi \subseteq RU\Phi$
(Prop. 7). For each of these inclusions, let us consider some counterexamples of the reverse one and discuss how we could improve the approximation. For that purpose, we will suppose that the six equalities $RIST_1(x,w) = IST_1(x,w)$, ... and $U\Phi(x,w) = RU\Phi(x,w)$ are true for any vertex w such that $deg_\Gamma(x,w) < deg_\Gamma(x,y)$ and see for what reasons they may not be true for (x,y).

1) Figure 3 provides a counterexample of the inclusion $RIST_1 \supseteq IST_1$: $y_1 \in IST_1(x,z')$, but $y_1 \notin RIST_1(x,z')$, as there is no path in Φ from x to y_1. This will happen to any vertex z in IST_1 such that the conclusion $x \rightarrow z$ is supported by different paths in the different credulous extension of Γ (called 'floating' conclusion by D.Makinson and K.Schlechta in [MS91]) because in that case z is not in $RIST_1$, as $\Phi^+(x,z) \subseteq I\Phi^+(x,z) = \emptyset$. Is there another reason of difference between $RIST_1$ and IST_1? We suppose that the six above equalities are true for any vertex w such that $deg_\Gamma(x,w) < deg_\Gamma(x,y)$. We have:

$z \in RIST_1$ iff ($\exists \alpha : x \stackrel{\alpha_1}{\rightarrow} z \rightarrow y, \alpha_1 \in \Phi$ and α is not preempted in $RU\Phi$)

$z \in RIST_1$ iff ($\exists \alpha : x \stackrel{\alpha_1}{\rightarrow} z \rightarrow y, \alpha_1 \in I\Phi$ and α is not preempted in $U\Phi$)

$z \in RIST_1$ iff (($\exists \alpha : x \stackrel{\alpha_1}{\rightarrow} z \rightarrow y, \alpha_1 \in I\Phi$) and not $P(z, U\Phi)$),
where $P(z, U\Phi)$ means: $\exists z', \alpha_1', \alpha_2'$ s.t. $z' \neq z$, $z' \rightarrow$

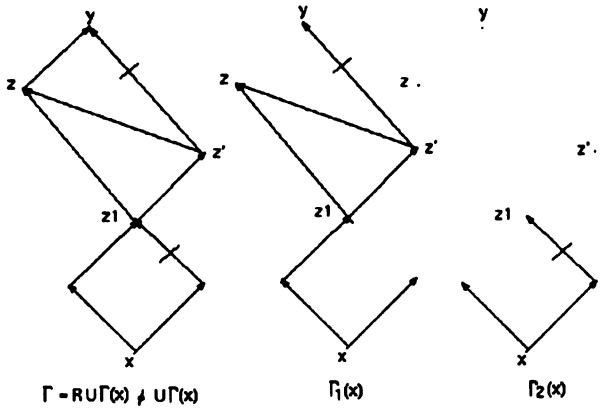


Figure 6: Insufficiency of $\Phi(\alpha)$

$\bar{y} \in U$ and $x \xrightarrow{\alpha'_1} z' \xrightarrow{\alpha'_2} z \in U\Phi$.
On the other hand:

$z \in IST_1$ iff ($\forall \Phi'$ credulous, $\exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in \Phi'$ and α is not preempted in Φ')

$z \in IST_1$ iff ($\forall \Phi'$ credulous, $\exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in \Phi'$ and not $P(z, \Phi')$)

$z \in IST_1$ iff (($\forall \Phi'$ credulous, $\exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in \Phi'$) and not $P(z, U\Phi)$)
Therefore, concerning IST_1, the introduction of an error comes from the problem of 'floating' conclusions and I can see no way of doing better.
(idem for ISF_1)

2) Figures 3 and 6 provide counterexamples of the inclusions $UST_1 \supseteq RUST_1$ and $USF_1 \supseteq RUSF_1$: in figure 3, $z' \in RUSF_1 - USF_1$, and in figure 6, where $\Gamma_1(x)$ and $\Gamma_2(x)$ are the subgraphs associated with the two credulous extensions of Γ, $z \in RUST_1 - UST_1$ (note that there is no 'floating' conclusion in figure 6). In both counterexamples, the error comes from the unsufficiency of the set $\Phi(\alpha)$. We suppose that the six equalities are true for any vertex w such that $deg_\Gamma(x, w) < deg_\Gamma(x, y)$. We have:

$z \in UST_1$ iff ($\exists \Phi'$ credulous, $\exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in \Phi'$ and α is not preempted in Φ')

$z \in UST_1$ iff ($\exists \Phi'$ credulous, ($\exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in \Phi'$) and not $P(z, \Phi')$)
On the other hand:

$z \in RUST_1$ iff ($\exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in RU\Phi$ and α is not preempted in $\Phi(\alpha)$)

$z \in RUST_1$ iff ($\exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y, \alpha_1 \in U\Phi$ and α is not preempted in $I\Phi(\alpha)$)

$z \in RUST_1$ iff ($\exists \alpha : x \xrightarrow{\alpha_1} z \rightarrow y$, ($\exists \Phi'$ credulous, $\alpha_1 \in \Phi'$) and not $P(z, I\Phi(\alpha))$)
Defining $\Phi(\alpha)$ so that $RUST_1$ approaches UST_1 as

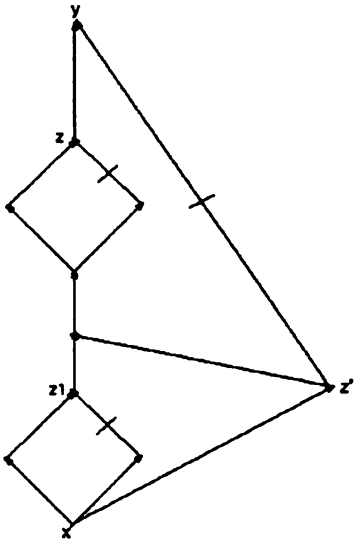


Figure 7: $\Phi(\alpha)$ is Better than α and Φ

closely as possible is not easy. To prove Proposition 7, we need the implication:

If the inclusions of Prop.7 are true for any vertex y' such that $deg_\Gamma(x, y') < deg_\Gamma(x, y)$ then

$\forall \Phi'$ credulous, $\forall \alpha : x \xrightarrow{\alpha_1} z \rightarrow y$, if $\alpha_1 \in \Phi'$ then $\Phi(\alpha) \subseteq \Phi'(x, z)$
$\Phi(\alpha)$ defined as the set $\{\alpha_1\}$, $\Phi^+(x, z)$ or as in this paper verifies above implication. The latter is the best one, because, as it contains the two others, it makes $RIST_1$ and Φ bigger and $RUST_1$ and $RU\Phi$ smaller, with inclusions of Prop.7 still true. For instance in figure 7, where $UST_1 = \emptyset$, if we define $\Phi(\alpha)$ as $\{\alpha_1\}$ or $\Phi^+(x, z)$ then $z \in RUST_1$, but not if we define it as in this paper. For the same reason, any set $\Phi_2(\alpha)$ verifying, under the same conditions as above:

$\Phi(\alpha) \subseteq \Phi_2(\alpha) \subseteq \Phi'(x, z)$
would come as an improvement of $\Phi(\alpha)$ as defined in this paper. For instance, let us define $\Phi_2(\alpha)$ as follows:

$\forall \alpha : x \xrightarrow{\alpha_1} z \rightarrow y$,

$\Phi_2(\alpha) = \{\mu_1 \in C^+_\Gamma(x, z) / \forall u : z' \rightarrow y'$ link of μ_1, u is a link of α_1 or $u \in U(x, y')$ or ($x \rightarrow \bar{y'} \notin U$ and $RUSF(x, y') = \emptyset$)$\}$
We have:

$\forall \Phi'$ credulous, $\forall \alpha' : x \xrightarrow{\alpha'_1} z' \rightarrow y'$, if $\alpha'_1 \in \Phi'$ and ($x \rightarrow \bar{y'} \notin U$ and $RUSF(x, y') = \emptyset$) and $SF(\Phi')(x, y') \subseteq RUSF(x, y')$ then $\alpha' \in \Phi'$
as α' verifies C.1), C.2) and C.3), and therefore:

If the inclusions of Prop.7 are true for any vertex y' such that $deg_\Gamma(x, y') < deg_\Gamma(x, y)$ then

$\forall \Phi'$ credulous, $\forall \alpha : x \xrightarrow{\alpha_1} z \rightarrow y$, if $\alpha_1 \in \Phi'$ then $\Phi(\alpha) \subseteq \Phi_2(\alpha) \subseteq \Phi'(x, z)$

Note that if we replaced $\Phi(\alpha)$ by $\Phi_2(\alpha)$, in figure 6, z would not be in $RUST_1$, because $\alpha : x \xrightarrow{\alpha'_1} z_1 \rightarrow z \rightarrow y$ is preempted in $\Phi_2(\alpha)$, as $\alpha' : x \xrightarrow{\alpha'_1} z_1 \rightarrow z' \rightarrow z$ is in $\Phi_2(\alpha) - \Phi(\alpha)$. But just adding vertex z_2 and links $z_1 \rightarrow z_2$ and $z_2 \rightarrow z'$ would make z fall into $RUST_1$ again. Moreover, though the definition of $\Phi_2(\alpha)$ does not look much more complex than that of $\Phi(\alpha)$, computing $RUST_1$ using $\Phi_2(\alpha)$ instead of $\Phi(\alpha)$ becomes NP-hard. This can be proved from a reduction from the NP-complete problem 'Path With Forbidden Pairs' (PWFP) defined by Gabow, Maheshwari and Osterweil in 1976 (see Selman et al. [SL91]).
Note that $RUST_1$-computing would still be NP-hard, replacing condition '$(x \rightarrow y' \notin U$ and $RUSF(x, y') = \emptyset)$' in the definition of $\Phi_2(\alpha)$ by simple condition 'there is no negative link into y' ' (same reduction as above). (idem for $RUSF_1$-computing)
It comes out of that reduction that the polynomiality of $RUST_1$-computing using $\Phi(\alpha)$ comes from the implication:

If $U^+(x, y') \neq \emptyset$ then $C^+_{\Gamma(x)}(x, y') \neq \emptyset$,

so that $\forall \alpha : x \xrightarrow{\alpha_1} z \rightarrow y$, if $\alpha' : x \xrightarrow{\alpha'_1} z' \xrightarrow{\alpha'_2} z \in \Phi(\alpha)$ and $z' \notin \alpha_1$ then $U^+(x, z') \neq \emptyset$, which entails $C^+_{\Gamma(x)}(x, z') \neq \emptyset$. (In the reduction, it reduces each 'forbidden pair' to a 'forbidden singleton'.) Therefore, in order to keep polynomial complexity, we could try to improve $\Phi(\alpha)$ by defining $\Phi_3(\alpha)$ verifying:

$\forall \alpha : x \xrightarrow{\alpha_1} z \rightarrow y$,

$\Phi_3(\alpha) = \{\mu_1 \in C^+_\Gamma(x, z) / \forall u : z' \rightarrow y'$ link of μ_1, u is a link of α_1 or $u \in U_3(x, y')\}$

and if $U^+_3(x, y') \neq \emptyset$ then $C^+_{\Gamma_3(x)}(x, y') \neq \emptyset$,
where $\Gamma_3(x) = (X, U_3(x))$, $U_3(x) = \cup\, U_3(x, y), y \in X$ and $\forall \Phi'$*credulous*, $\forall (x, y) \in X^2$, $U(x, y) \subseteq U_3(x, y) \subseteq U_{\Phi'}(x, y)$
I can see no way of doing better than $\Phi(\alpha)$.

3) Figure 3 provides a counterexample of the inclusion $\Phi \supseteq I\Phi$: $\alpha : x \rightarrow z \rightarrow y \in I\Phi - \Phi$. As we have: $\Phi(x) = C_{\Gamma(x)}(x)$ and $I\Phi(x) = C_{I\Gamma(x)}(x)$, the difference between Φ and $I\Phi$ comes from that between the sets $U(x, y)$ and $IU(x, y)$, i.e. between the sets $RIST[F]_1$ and $IST[F]_1$ on one hand, and $RUST[F]_1$ and $UST[F]_1$ on the other hand, which has been studied in 1) and 2). Note that if he six equalities are true for any vertex w such that $deg_\Gamma(x, w) < deg_\Gamma(x, y)$ and $\alpha : x \xrightarrow{\alpha_1} z \rightarrow y \in I\Phi - \Phi$ then $z \in RIST_1$ and the error comes from the non-emptyness of $RUSF_1$ (called 'floating' acceptance by D.Makinson and K.Schlechta in [MS91]).

4) Figures 3 and 6 provide counterexamples of the inclusion $U\Phi \supseteq RU\Phi$: in figure 3, $\alpha' : x \rightarrow z' \rightarrow \bar{y} \in RU\Phi - U\Phi$, and in figure 6, $\alpha : x \xrightarrow{\alpha'_1} z_1 \rightarrow$

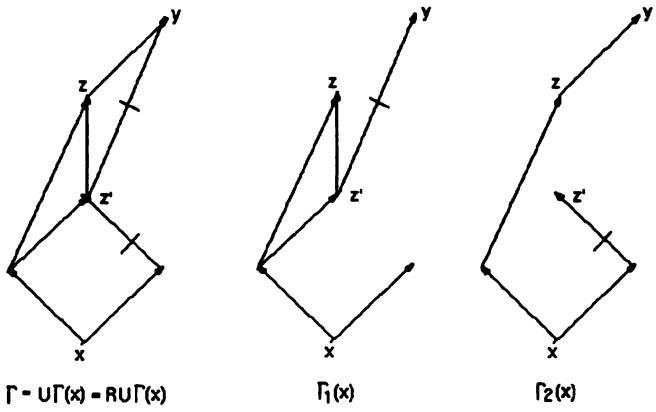


Figure 8: A Path in $U\Gamma(x)$ is Not Always in $U\Phi(x)$

$z \rightarrow y \in RU\Phi - U\Phi$. As we have: $U\Phi(x) \subseteq C_{U\Gamma(x)}(x)$ and $RU\Phi(x) = C_{RU\Gamma(x)}(x)$, the difference between $C_{U\Gamma(x)}(x)$ and $RU\Phi(x)$ comes from that between the sets $UU(x, y)$ and $RUU(x, y)$, i.e. between the sets $UST[F]_1$ and $RUST[F]_1$, which has been studied in 2) (called 'floating' defeat by D.Makinson and K.Schlechta in [MS91]). A difference may be introduced from that between $U\Phi(x)$ and $C_{U\Gamma(x)}(x)$. For instance, in figure 8, where $\Gamma_1(x)$ and $\Gamma_2(x)$ are the subgraphs associated with the two credulous extensions of Γ, path $\alpha : x \xrightarrow{\alpha'_1} z' \rightarrow z \rightarrow y \in C_{U\Gamma(x)}(x) - U\Phi(x)$. In that case, there is no graph whose set of paths from x is equal to $U\Phi(x)$, and I cannot see how we could do better than approaching $U\Phi(x)$ by $C_{RU\Gamma(x)}(x)$, within polynomial complexity.

Concerning the difference between the conclusion sets C and IC, and UC and RUC, we introduce an error by approaching IC by $C(\Phi)$, as $C(I\Phi) \subseteq IC$, with strict inclusion in case of 'floating' conclusion, but not by approaching UC by $C(RU\Phi)$, as $UC = C(U\Phi)$.

6.5 DISCUSSION

I showed in section 5 that searching a conclusion set equal to the intersection of the conclusion sets of the credulous extensions of Γ is intuitively correct (Prop.2), but untractable (Prop.4) and cannot be obtained from an extension in a natural way (Prop.5).
I defined RS theory, verifying the inclusion:

$$C(\Phi) \subseteq \bigcap_{\Phi' \text{ credulous}} C(\Phi')$$

(Prop.8), which is the reverse inclusion of the one verified if Φ is the skeptical extension of Γ (Prop.3).
RS theory is therefore preferable to skeptical one *whenever it is important to adopt a really skeptical attitude*, i.e. whenever it is better not to draw a 'valid' conclusion (in the sense of 'drawn in any credulous extension of Γ'), than to draw an 'invalid' one (see poisonous mushrooms interpretation of figure 1 in section 4).

We know from Prop.4 that IC, UC,$I\Phi$ and $U\Phi$-computing problems are NP-hard. What about the complexity of UST_1 and IST_1-computing problems? They are proved to be NP-hard from the reduction from $IC(x,y)$ and $UC(x,y)$-computing ones defined in the proof of Proposition 4. In that reduction we have:

$x \rightarrow y \in IC$ in Γ iff $z_1 \notin UST_1$ in Γ'

$x \rightarrow y \in UC$ in Γ iff $z_1 \notin IST_1$ in Γ'

(The NP-hardness of UST_1-computing problem can also be proved from that of UC-computing one and the equivalence:

$x \rightarrow y \in UC$ iff ($x \rightarrow y \in U$ or $UST_1 \neq \emptyset$)

7 CONCLUSION

RS theory is really skeptical, as it only draws conclusions that are drawn in every credulous extension of Γ.It is polynomially tractable, and provides an heuristics to compute the ideal skeptical conclusion set of Γ, defined as the intersection of credulous ones, which is untractable. It remains to evaluate and improve its performance by experiments and further research. At a higher level, is the definition of a credulous extension of Γ chosen in this paper (construction by upward concatenation, off-path without reinstatement preemption) the best one? And for which graphs? Inheritance graphs are diverse and may need different treatments.

References

[Bre87] G. Brewka. The logic of inheritance in frame systems. *IJCAI-87*, 1987.

[Gre90a] E. Gregoire. ***Logiques non monotones et intelligence artificielle.*** Hermes, Paris, 1990.

[Gre90b] E. Gregoire. Skeptical inheritance can be more expressive. *ECAI-90*, pages 326–332, 1990.

[Hor91a] J.F. Horty. A credulous theory of mixed inheritance. In M.Lenzerini, D.Nardi, and M.Simi, editors, *Inheritance Hierarchies in Knowledge Representation and Programming Langages*, chapter 2. John Wiley, 1991.

[Hor91b] J.F. Horty. Some direct theories of nonmonotonic inheritance. In D.Gabbay and C.Hogger, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming.* Oxford University Press, 1991. (Forthcoming in 1991).

[HTT90] J.F. Horty, R.H. Thomason, and D.S. Touretzky. A skeptical theory of inheritance in nonmonotonic semantic networks. *Artificial Intelligence*, 42:311–348, 1990.

[MS91] D. Makinson and K. Schlechta. Floating conclusions and zombie paths: two deep difficulties in the "directly skeptical" approach to defeasible inheritance nets. *Artificial Intelligence*, 48:199–209, 1991.

[Poo85] D.L. Poole. On the comparison of theories: Preferring the most specific explanation. *IJCAI-85*, pages 144–147, 1985.

[Ryc89] P. Rychlik. Multiple inheritance systems with exceptions. *Artificial Intelligence*, 3:159–176, 1989.

[San86] E. Sandewall. Nonmonotonic inference rules for multiple inheritance with exceptions. *IEEE 74*, pages 1345–1353, 1986.

[Sch90] K. Schlechta. Directly skeptical inheritance cannot capture the intersection of extensions. *Sankt Augustin Workshop on Nonmonotonic Reasoning*, 1990.

[Sch92] K. Schlechta. Results on non-monotonic logics. Technical Report IWBS Report 204, IBM Stuttgart, January 1992. POB 800880, 7000 Stuttgart 80, Germany.

[Sim91] G. Simonet. Le problème des exceptions dans un graphe d'héritage multiple. D.E.A. report, 1991.

[SL91] B. Selman and H.J. Levesque. The tractability of path-baed inheritance. In M.Lenzerini, D.Nardi, and M.Simi, editors, *Inheritance Hierarchies in Knowledge Representation and Programming Langages*, chapter 6. John Wiley, 1991.

[Ste89] L.A. Stein. Skeptical inheritance: Computing the intersection of credulous extensions. *IJCAI-89*, 2:1153–1158, 1989.

[Ste91] L.A. Stein. Computing skeptical inheritance. In M.Lenzerini, D.Nardi, and M.Simi, editors, *Inheritance Hierarchies in Knowledge Representation and Programming Langages*, chapter 5. John Wiley, 1991.

[TH89] R.H. Thomason and J.F. Horty. Logics for inheritance theory. *Artificial Intelligence*, 346:220–237, 1989.

[THT87] D.S. Touretzky, J.F. Horty, and R.H. Thomason. A clash of intuitions: the current state of nonmonotonic multiple inheritance systems. *IJCAI-87*, 1:476–482, 1987.

[Tou86] D.S. Touretzky. ***The Mathematics of Inheritance Systems.*** Pitman, London, 1986.

[TTH91] D.S. Touretzky, R.H. Thomason, and J.F. Horty. A skeptic's menagerie: Conflictors, preemptors, reinstaters and zombies in nonmonotonic inheritance. *IJCAI-91*, 1:478–483, 1991.

On the Impact of Stratification on the Complexity of Nonmonotonic Reasoning

Ilkka Niemelä
Department of Computer Science
Helsinki University of Technology
Otakaari 1, SF-02150 ESPOO, Finland

Jussi Rintanen
Department of Computer Science
Helsinki University of Technology
Otakaari 1, SF-02150 ESPOO, Finland

Abstract

The ability to "define" propositions using default assumptions about the same propositions is identified as a source of additional computational complexity in nonmonotonic reasoning. If such constructs are not allowed, i.e. the knowledge base is stratified, a significant computational advantage is obtained. This is demonstrated by developing an iterative algorithm for propositional stratified autoepistemic theories the complexity of which is dominated by required classical reasoning. Thus efficient subclasses of stratified nonmonotonic reasoning can be obtained by further restricting the form of sentences in the knowledge base. As an example we derive quadratic and linear time algorithms for specific subclasses of stratified autoepistemic theories. The results are shown to imply efficient reasoning methods for stratified cases of default logic, logic programs, truth maintenance systems, and nonmonotonic modal logics.

1 INTRODUCTION

Nonmonotonic reasoning is an important aspect of many knowledge representation systems. Nonmonotonic reasoning is applied because it is hoped that knowledge representation and reasoning problems can be solved more effectively than using classical (monotonic) reasoning. Recent results suggest that nonmonotonic reasoning is in fact computationally more complex than corresponding classical reasoning [Gottlob, 1991]. Furthermore, even very restricted subclasses of nonmonotonic reasoning where required classical reasoning can be done efficiently turn out to be computationally intractable. Examples of this are, e.g., simple cases of default logic [Kautz and Selman, 1991], truth maintenance systems [Elkan, 1990], and logic programs [Marek and Truszczyński, 1991] which have NP-complete decision problems. Thus reducing the computational complexity of required classical reasoning does not yield the expected efficiency in nonmonotonic reasoning. A typical aspect of the subclasses of nonmonotonic reasoning with disappointing computational properties is that propositions can be "defined" in terms of default assumptions about the same propositions. This seems to result in a situation where finding the correct order of applying defaults (conflict resolution) is computationally very complex.

In this paper we investigate stratified knowledge bases. The notion of stratification has its origins in the logic programming community [Chandra and Harel, 1985; Apt *et al.*, 1988; Van Gelder, 1988]. A knowledge base is stratified if it can be partitioned into a set of levels (strata) such that on every level default assumptions are made only about propositions which have already been "defined" on lower levels. This restriction reduces expressivity at least in the sense that it rules out multiple extensions: a stratified knowledge base has exactly one possible set of correct conclusions. However, we show that the restriction provides notable computational benefits. In stratified knowledge bases the conflict resolution task can be solved efficiently and the overall complexity of reasoning is dominated by the complexity of the classical reasoning task.

Stratified propositional autoepistemic theories [Marek and Truszczyński, 1991] are chosen as the basis of the research because autoepistemic logic offers a unified approach to several other types of nonmonotonic reasoning (or at least substantial fragments of these) [Konolige, 1988; Gelfond and Lifschitz, 1988; Elkan, 1990]. Thus efficient decision methods developed for autoepistemic logic can be immediately applied to other forms of nonmonotonic reasoning. First we develop an iterative algorithm for reasoning in stratified autoepistemic theories. From this we derive a quadratic and a linear time algorithm for limited subclasses of stratified theories. The result are shown to imply fast reasoning algorithms for stratified cases of default logic, logic programs, truth maintenance systems, and nonmonotonic modal logics.

2 AUTOEPISTEMIC LOGIC

To obtain the language $\mathcal{L}_{ae}$ of propositional autoepistemic logic we extend the language $\mathcal{L}$ of the propositional calculus by a monadic operator L which is read "is believed". Autoepistemic logic models the beliefs of a fully introspective ideally rational agent. The agent reasons according to a consequence relation $\models$ which is a simple extension of the propositional consequence relation where the new $L\phi$ formulae are treated like atomic formulae in the propositional calculus. Given a set of premises a set of correct autoepistemic conclusions is defined as a set of beliefs of the agent with the premises as the initial assumptions of the agent. A set of beliefs is called a stable expansion of the premises and it is defined by the following fixed point equation.

Definition 2.1 (Moore [1985]) *Δ is a* stable expansion *of Σ iff*

$$\Delta = \{\phi \mid \Sigma \cup L\Delta \cup \neg L\overline{\Delta} \models \phi\} \qquad (1)$$

where $L\Delta = \{L\phi \mid \phi \in \Delta\}$, $\neg\Delta = \{\neg\phi \mid \phi \in \Delta\}$, and $\overline{\Delta} = \mathcal{L}_{ae} - \Delta$. Thus $\neg L\overline{\Delta} = \{\neg L\phi \mid \phi \in \mathcal{L}_{ae} - \Delta\}$.

Stable expansions are infinite sets of formulae. A finitary characterization is needed for handling expansions computationally. Niemelä [1990] has presented a compact characterization of stable expansions using the notion of a *full set*. The basic idea is to use the $L\phi$ subformulae of the premises to characterize the stable expansions. If the set of premises is finite, the corresponding full sets are finite. In this case infinite stable expansions can be represented finitely.

We use the following notations. $Sf^L(\phi)$ denotes the set of subformulae of the form $L\chi$ of ϕ. $Sf^{qL}(\phi)$ is the set of $L\chi$ quasi-subformulae of ϕ. Quasi-subformulae are subformulae in the usual sense except that $L\chi$ formulae do not have any further quasi-subformulae. For a set of formulae Σ, $Sf^L(\Sigma) = \bigcup_{\phi\in\Sigma} Sf^L(\phi)$ and similarly for $Sf^{qL}(\Sigma)$.

Definition 2.2 *A set of formulae Λ is Σ-full if it satisfies the following conditions.*

1. $\Lambda \subseteq Sf^L(\Sigma) \cup \neg Sf^L(\Sigma)$.
2. $L\phi \in \Lambda$ *iff* $\Sigma \cup \Lambda \models \phi$ *for all* $L\phi \in Sf^L(\Sigma)$.
3. $\neg L\phi \in \Lambda$ *iff* $\Sigma \cup \Lambda \not\models \phi$ *for all* $L\phi \in Sf^L(\Sigma)$.

For a set of premises Σ, the Σ-full sets are in a one-to-one correspondence with the stable expansions of Σ [Niemelä, 1990]. The unique stable expansion induced by a full set can be characterized with the aid of the consequence relation $\models_L$ which is defined recursively using the underlying consequence relation $\models$. The new consequence relation determines the membership in a stable expansion of Σ when the corresponding full set Λ is known (Definition 4.1 and Theorems 3.15 and 4.2 [Niemelä, 1990]):

Definition 2.3 *For $\Sigma \subseteq \mathcal{L}_{ae}$ and $\phi \in \mathcal{L}_{ae}$,*

$$\Sigma \models_L \phi \text{ iff } \Sigma \cup SB_\Sigma(\phi) \models \phi$$

where $SB_\Sigma(\phi) = \{L\chi \in Sf^{qL}(\phi) \mid \Sigma \models_L \chi\} \cup \{\neg L\chi \in \neg Sf^{qL}(\phi) \mid \Sigma \not\models_L \chi\}$.

Theorem 2.4 *Let Λ be a Σ-full set. Then $\Delta = \mathrm{SE}_\Sigma(\Lambda) = \{\phi \mid \Sigma \cup \Lambda \models_L \phi\}$ is the unique stable expansion of Σ such that $\Lambda \subseteq L\Delta \cup \neg L\overline{\Delta}$.*

Example 1 Let $\Sigma = \{Lp \rightarrow p, \neg Lp \rightarrow q\}$, where p and q are atomic. There are two candidates for Σ-full sets: $\Lambda_1 = \{Lp\}$ and $\Lambda_2 = \{\neg Lp\}$. Both are Σ-full. Λ_1 is full as $\Sigma \cup \Lambda_1 \models p$ and Λ_2 is full as $\Sigma \cup \Lambda_2 \not\models p$. So Σ has exactly two stable expansions $\mathrm{SE}_\Sigma(\{Lp\})$ and $\mathrm{SE}_\Sigma(\{\neg Lp\})$. $L\neg Lq$ belongs to the former but not to the latter because $\Sigma \cup \{Lp\} \models_L L\neg Lq$ but $\Sigma \cup \{\neg Lp\} \not\models_L L\neg Lq$. E.g., $\Sigma \cup \{Lp\} \models_L L\neg Lq$ can be verified as follows. As $\mathrm{SB}_{\Sigma\cup\{Lp\}}(q) = \emptyset$ and $\Sigma\cup\{Lp\} \not\models q$, $\Sigma\cup\{Lp\} \not\models_L q$. Thus $\mathrm{SB}_{\Sigma\cup\{Lp\}}(\neg Lq) = \{\neg Lq\}$. So $\Sigma \cup \{Lp\} \cup \mathrm{SB}_{\Sigma\cup\{Lp\}}(\neg Lq) \models \neg Lq$ which implies $\Sigma\cup\{Lp\} \models_L \neg Lq$. Hence $\mathrm{SB}_{\Sigma\cup\{Lp\}}(L\neg Lq) = \{L\neg Lq\}$ and thus $\Sigma\cup\{Lp\} \models_L L\neg Lq$. ■

3 STRATIFICATION

Marek and Truszczyński define a notion of a stratified set for propositional autoepistemic logic.

Definition 3.1 ([Marek and Truszczyński, 1991]) *A set of formulae Σ is stratified if*

1. *the formulae $\phi \in \Sigma$ are of the form $a(\phi) \wedge o(\phi) \rightarrow c(\phi)$, where the subformulae $o(\phi)$ and $c(\phi)$ do not contain the operator L and $o(\phi)$ may be missing, and $a(\phi)$ is a formula of the form $L\phi_1 \wedge \cdots \wedge L\phi_r \wedge \neg L\psi_1 \wedge \cdots \wedge \neg L\psi_s$ where $r, s \geq 0$.*
2. *The set $\{c(\phi) | \phi \in \Sigma\}$ is satisfiable.*
3. *There exists a set of indices $I = \{1, \ldots, n\}$ or $I = \{1, \ldots\}$ and a partition of $\Sigma = \bigcup_{i\in I} \Sigma_i$ such that for all $j \in I$ the propositional variables occurring in $\{c(\phi) | \phi \in \Sigma_j\}$ do not occur in Σ_j in the scope of an L operator or in Σ_1^{j-1} (where $\Sigma_a^b = \bigcup_{i=a}^b \Sigma_i$).*

If a set of premises is stratified, it has a unique stable expansion [Marek and Truszczyński, 1991, Theorem 5.1]. However, a non-stratified set can have several stable expansions. In fact there can be up to 2^n stable expansions for a set of premises with n $L\chi$ subformulae [Niemelä, 1990]. On the other hand, it is not necessary for a set to be stratified to have a unique stable expansion. The set $\{p, \neg Lp \rightarrow p\}$ is a simple example of this.

The notion of stratifiedness reduces expressivity because it rules out premises for which the agent has

multiple competing sets of beliefs. When formalizing commonsense reasoning premises with multiple stable expansions seem to occur (e.g. [Gelfond, 1988]). Kolaitis [1991] discusses the expressivity of stratified logic programs which are closely related to stratified autoepistemic theories. Stratification can be seen as an attempt to find a useful trade-off between the expressivity of a knowledge representation language and the computational complexity of the required reasoning effort. On one hand, stratification excludes multiple expansions and reduces expressivity and, on the other hand, it leads to efficient nonmonotonic reasoning methods as will be shown in this paper.

The reasoning methods developed in this paper use the stratifications, i.e. the partitions of the formulae in the stratified sets. In the general case the computation of a stratification or testing whether a set of autoepistemic formulae is stratified is exponential time because of the satisfiability testing of $\{c(\phi)|\phi \in \Sigma\}$. In the polynomial time cases developed in this paper Condition 2 of stratification can be tested in linear time. For testing Condition 3 efficiently Marek and Truszczyński define the notion of *a-stratifiedness* which coincides with stratifiedness. A finite set Σ is a-stratified if it satisfies conditions 1 and 2 of stratification and there is a partition $P_1, \ldots, P_n$ of the propositional variables P in Σ that fulfills the following condition. For each pair of variables $p \in P_i, q \in P_j$ such that p occurs in $a(\phi)$ and q in $c(\phi)$ for some $\phi \in \Sigma$, $i < j$, and for each pair of variables $p \in P_i, q \in P_j$ such that p occurs in ϕ and q in $c(\phi)$ for some $\phi \in \Sigma$, $i \leq j$.

Theorem 3.2 ([Marek and Truszczyński, 1991]) *A finite set of formulae Σ is stratified if and only if Σ is a-stratified.*

The test for the existence of the partition of variables can easily be reduced to the computation of the strong components of a graph. The *characteristic graph* of a set of formulae is a graph representing the constraints imposed on the partition of the propositional variables by the definition of a-stratifiedness. In the characteristic graph there is an edge from the variable p to the variable q if for some $\phi \in \Sigma$ p occurs in $c(\phi)$ and q anywhere in ϕ. The edge is an L-edge if there is an occurrence of q in $a(\phi)$.

Theorem 3.3 ([Marek and Truszczyński, 1991]) *A finite set of formulae Σ is a-stratified if and only if Σ satisfies Conditions 1 and 2, and no strong component of its characteristic graph contains an L-edge.*

Example 2 The set

$$r \wedge \neg Lp \rightarrow h$$
$$q \wedge \neg Lh \rightarrow p$$

is not a-stratified and consequently it is not stratified. The characteristic graph has three strong components. The variables r and p occupy singleton strong components, and since there are edges both from h to p and from p to h, p and h are in the same strong component. The set is not a-stratified – and consequently not stratified – because the edges between p and h are L-edges. ■

Example 3 The set

$$L\neg Lr \wedge p \rightarrow q$$
$$q \rightarrow p$$
$$L(p \leftrightarrow q) \rightarrow s$$

is a-stratified. Variables p and q belong to the same component and there are no L-edges between them, s and r both occupy a singleton component. ■

For the tractable classes of autoepistemic reasoning investigated in this paper, we present a linear time algorithm that computes a stratification if the set is stratified, and for sets that are not, detects this fact. The algorithm is based on Theorem 3.3. Marek and Truszczyński [1991] sketch a similar algorithm for arbitrary sets of autoepistemic formulae that fulfill Condition 1. Another algorithm for computing a stratification which is based on strong components is presented in [Lassez *et al.*, 1987].

In the general case the size of the characteristic graph is quadratic on the size of Σ, and consequently the traversal of the graph for finding the strong components takes quadratic time. However, if the form of the subformulae $c(\phi), \phi \in \Sigma$ is restricted the computation becomes linear time.

Linear time complexity results in this paper, e.g. the linearity of our stratification algorithm, rest on the following assumption.

Assumption 3.4 *Each propositional variable is assigned a unique number so that data structures with constant access time (arrays) can be used for storing various data related to them.*

Taking a set of formulae Σ as input and assigning a number for each propositional variable can be done in $\mathcal{O}(n \log v)$ time where n is the size of Σ and v is the number of distinct propositional variables in Σ.

Proposition 3.5 *Let Σ be a set of formulae that fulfills Condition 1 of stratification, and for each $\phi \in \Sigma$ there are occurrences of at most one propositional variable in $c(\phi)$. Under Assumption 3.4 the computation of a stratification for Σ or detecting that Σ is not stratified is $\mathcal{O}(|\Sigma|)$ time.*

In this restricted case Condition 2 of the definition of a stratified set can be tested in time $\mathcal{O}(|\Sigma|)$.

By Theorem 3.3 testing Condition 3 of stratification can be implemented as a computation of the strong components of the characteristic graph together with detection of L-edges inside the strong components. For this purpose we give a variant of Tarjan's [1972] well-known algorithm for the strong components of a graph as presented in [Aho *et al.*, 1974]. Tarjan's algorithm runs in time $\mathcal{O}(n+e)$ where n is the number of nodes and e is the number of edges, and it has the useful property that the strong components are produced in an order that qualifies as a stratification, i.e. the first component the algorithm emits consists of the variables in $c(\phi)$ for formulae $\phi \in \Sigma_1$ that can be taken as the lowest stratum of a stratification Σ_1^n, and so on.

The size of the characteristic graph is linear on the size of Σ because for each $\phi \in \Sigma$ there are occurrences of at most one propositional variable in $c(\phi)$ and consequently there is at most one edge for each variable occurrence in $\{a(\phi) \wedge o(\phi) | \phi \in \Sigma\}$.

For the computation of the strata the following arrays and variables are needed.

formulae[p] the list of formulae ϕ in which the variable p appears in $c(\phi)$. This array can be initialized in linear time by traversing the set of formulae once.

edges[p] the list of variables q that appear in $a(\phi) \wedge o(\phi)$ for a formula ϕ in which the variable p appears in $c(\phi)$. The initialization can be done in linear time. First initialize the elements to empty lists. Then for each p the list of formulae [p] is traversed and for each occurrence of a variable q an auxiliary array of flags is tested whether q already is in edges[p]. If not, it is added in the head and the auxiliary array is updated.

l-edges[p] the list of variables q that appear in $a(\phi)$ for a formula ϕ in which the variable p appears in $c(\phi)$. This array is initialized the same way as **edges**.

ccount a counter for the strong components. Initialized to zero.

component[i] the list of formulae in stratum i.

stratum[p] the number of the stratum to which formulae ϕ having p in $c(\phi)$ belong.

The following variables are part of the original strong components algorithm (see [Aho *et al.*, 1974] for details):

count a counter for numbering the nodes of the graph in the order of depth-first traversal.

dfnumber[p] the number assigned to the node p during depth-first traversal.

lowlink[p] an array which is used for recognizing strong components during the traversal.

SEARCHC is called repeatedly for unmarked variables until all variables are marked *old*. If the error NOT_STRATIFIED is signalled then Condition 3 cannot be fulfilled and the set is not stratified.

```
procedure SEARCHC(v);
begin
  mark v "old"
  dfnumber[v] := count;
  count := count + 1;
  lowlink[v] := dfnumber[v];
  push v on stack;
  for each vertex w on edges[v] do
   if w is marked "new" then
    begin
     SEARCHC(w);
     lowlink[v] := min(lowlink[v],lowlink[w]);
    end
   else
    if dfnumber[w] < dfnumber[v] and w is on stack
     then lowlink[v] := min(dfnumber[w],lowlink[v]);
   end if
  end for
  if lowlink[v] = dfnumber[v] then
   begin
    ccount := ccount + 1;
    components[ccount] := empty_list;
    initialize stack2 to empty;
    repeat
     pop x from top of stack;
     push x to stack2;
     stratum[x] := ccount;
    until x=v;
    while stack2 not empty do
     pop x from stack2;
     for each y in l-edges[x] do
      if stratum[x] = stratum[y]
       then signal NOT_STRATIFIED;
     end for
     concatenate formulae[x] to components[ccount];
    end while
   end
end
```

The first half of the algorithm traverses the characteristic graph depth-first and maintains the data structures for detection of the strong components. Like Tarjan's original algorithm it works in $\mathcal{O}(n+e)$ time where n and e are the number of nodes and edges, respectively. All our modifications are in the *if* statement that forms the second half of the algorithm.

First, for the formulae in the new component a new element is reserved in the array *components*. The *repeat* loop assigns each variable in the component the number of the component. Finally the *while* loop tests the component for the containment of an L-edge and concatenates the list of formulae belonging to the stratum to the array *components*. Constructing the lists

of the array *components* is $\mathcal{O}(n)$ where n is the size of the set of formulae, since each formula belongs to exactly one stratum and we restrict $c(\phi)$ for each ϕ to contain occurrences of at most one propositional variable. The tests for the containment of an edge through an L operator are within the $\mathcal{O}(n)$ bound as for each propositional variable p the presence of L-edges inside the stratum of p is tested exactly once, and the number of elements in the lists of the array *l-edges* is linearly bounded by the size of the set of formulae.

Example 4 Our algorithm computes for the set in Example 3 the stratification shown below. The first column contains the numbers of the strata, the second contains the variables in the corresponding strong components of the associated characteristic graph, and the third the strata, i.e. the sets of formulae ϕ in which the variables of the respective strong component occur in $c(\phi)$.

3	s	$L(p \leftrightarrow q) \rightarrow s$
2	p, q	$q \rightarrow p, L\neg Lr \wedge p \rightarrow q$
1	r	$\emptyset$

Because there are no formulae ϕ with r in $c(\phi)$ the lowest stratum is empty, and can be ignored. ■

4 A DECISION PROCEDURE FOR STRATIFIED THEORIES

The next theorem gives an iterative algorithm for computing the Σ-full set of an arbitrary stratified set Σ. Because of the results of Marek and Truszczyński and on the other hand of Niemelä, the Σ-full set characterizes the unique stable expansion of Σ.

Theorem 4.1 *Let $\Sigma = \Sigma_1^n$ be a stratified set. Then $\Lambda = \Lambda_n$ defined by*

$$\Lambda_0 = \emptyset$$
$$\Lambda_{i+1} = \Lambda_i \cup \{L\chi | L\chi \in Sf^L(\Sigma_{i+1}) - Sf^L(\Sigma_1^i), \Sigma_1^i \cup \Lambda_i \models_L \chi\} \cup \{\neg L\chi | L\chi \in Sf^L(\Sigma_{i+1}) - Sf^L(\Sigma_1^i), \Sigma_1^i \cup \Lambda_i \not\models_L \chi\}$$

$(0 \leq i < n)$ is Σ-full and $SE_\Sigma(\Lambda)$ is the unique stable expansion of Σ.

For proving Theorem 4.1 the following lemma is essential.

Lemma 4.2 *Let $\Sigma = \Sigma_1^n$ be stratified and Λ_i as in Theorem 4.1. Then for all $i, 0 \leq i < n$ and for all $L\chi \in Sf^L(\Sigma_{i+1})$, $\Sigma_1^i \cup \Lambda_i \models_L \chi$ iff $\Sigma_1^j \cup \Lambda_j \models_x \chi$ where $\models_x$ is $\models$ or $\models_L, i < j \leq n$.*

Proof: By induction on i. When $i = 0, \Sigma_1^0 = \emptyset$ and $\Lambda_0 = \emptyset$. Otherwise as in the inductive case.

$(i \geq 0)$ We prove $\Sigma_1^i \cup \Lambda_i \models_L \chi$ iff $\Sigma_1^j \cup \Lambda_j \models_x \chi$ by induction on the L-depth s of χ. The L-depth of a formula is the maximum nesting level of L operators in it. The proof for the base case $s = 0$ is included in the proof of the inductive case as the only difference is that when $s = 0$ the sets $SB_{\Sigma_1^k \cup \Lambda_k}(\chi)$ for $k \in \{i, j\}$ are empty.

We have to show that for all $L\chi \in Sf^L(\Sigma_{i+1})$, $\Sigma_1^i \cup \Lambda_i \cup SB_{\Sigma_1^i \cup \Lambda_i}(\chi) \models \chi$ iff $\Sigma_1^j \cup \Lambda_j \models_x \chi, i < j \leq n$. First note that **A.** $SB_{\Sigma_1^i \cup \Lambda_i}(\chi) = SB_{\Sigma_1^j \cup \Lambda_j}(\chi)$ by the induction hypothesis on s and the definition of $SB_\Sigma(\chi)$. **B.** $SB_{\Sigma_1^i \cup \Lambda_i}(\chi) \subseteq \Lambda_{i+1}$ (and furthermore $SB_{\Sigma_1^i \cup \Lambda_i}(\chi) \subseteq \Lambda_h$ for $h > i$) because for each $L\phi \in SB_{\Sigma_1^i \cup \Lambda_i}(\chi)$ there is $k \leq i$ such that $L\phi \in (Sf^L(\Sigma_{k+1}) - Sf^L(\Sigma_1^k))$ and by the induction hypothesis on i, $\Sigma_1^k \cup \Lambda_k \models_L \phi$, and by the equations of Theorem 4.1 $L\phi \in \Lambda_{k+1}$. Similarly for $\neg L\phi \in SB_{\Sigma_1^i \cup \Lambda_i}(\chi)$. By **A** and the monotonicity of $\models$ we get the implication $\Rightarrow$ for $\models_L$, and by monotonicity and **B** for $\models$.

$(\Leftarrow)$ We show that $\Sigma_1^i \cup \Lambda_i \not\models_L \chi$ implies $\Sigma_1^j \cup \Lambda_j \not\models_x \chi$. Let $\mathcal{M}$ be a model such that $\mathcal{M} \models \Sigma_1^i \cup \Lambda_i \cup SB_{\Sigma_1^i \cup \Lambda_i}(\chi)$ and $\mathcal{M} \not\models \chi$. Let $\mathcal{M}'$ be a model constructed by modifying $\mathcal{M}$ to satisfy $\Sigma_{i+1}^j \cup (\Lambda_j - \Lambda_i)$. This modification is possible without falsifying Σ_1^i because the set of propositional variables in $\{c(\phi) | \phi \in \Sigma_{i+1}^j\}$ is both disjoint from the propositional variables in $\Sigma_1^i \cup \{\chi\}$ and satisfiable according to the definition of stratification. Hence, $\Sigma_1^j \cup \Lambda_j \not\models \chi$. Although $SB_{\Sigma_1^i \cup \Lambda_i}(\chi)$ is not disjoint from Λ_j, it is contained in it because of **B** above. Then by **A** $SB_{\Sigma_1^j \cup \Lambda_j}(\chi) \subseteq \Lambda_j$ and therefore $\Sigma_1^j \cup \Lambda_j \not\models_L \chi$. □

Proof of Theorem 4.1: Λ is Σ-full because it satisfies the conditions of Definition 2.2. Condition 1 is immediate, and Condition 3 is true if Condition 2 is, since by construction for all $L\chi \in Sf^L(\Sigma)$ exactly one of $L\chi$ or $\neg L\chi$ is in Λ. Condition 2 holds because $L\chi \in \Lambda$ iff there is $i < n$ for which $L\chi \in (Sf^L(\Sigma_{i+1}) - Sf^L(\Sigma_1^i))$ and $\Sigma_1^i \cup \Lambda_i \models_L \chi$ and by Lemma 4.2 $\Sigma_1^n \cup \Lambda \models \chi$. By Theorem 5.1 of [Marek and Truszczyński, 1991] and Theorem 2.4 $SE_\Sigma(\Lambda)$ is the unique stable expansion of Σ. □

The following two lemmata are needed for Theorem 4.5 which shows how the unique stable expansion of a stratified set Σ can be computed more efficiently without explicitly constructing the Σ-full set Λ.

Lemma 4.3 *Let Σ be a set of formulae of the form $L\chi_1 \wedge \cdots \wedge L\chi_n \wedge \neg L\chi_{n+1} \wedge \cdots \wedge \neg L\chi_{n+m} \wedge \psi \rightarrow \psi'$ where $\psi, \psi' \in \mathcal{L}$, and let Λ be a set of formulae of the form $L\phi, \neg L\phi$ that contains exactly one of $L\chi$, $\neg L\chi$ for each $L\chi \in Sf^L(\Sigma)$. Define $\mathrm{Red}(\Sigma, \Lambda) = \{\psi \rightarrow \psi' | (\phi \wedge \psi \rightarrow \psi') \in \Sigma$ and the conjuncts of ϕ are in $\Lambda\}$. Assume that for all $L\phi, \neg L\phi' \in \Lambda$, $\Sigma \cup \Lambda \models_L \phi$ and*

$\Sigma \cup \Lambda \not\models_L \phi'$. *Then for all* $\chi \in \mathcal{L}_{ae}$,

$$\mathrm{Red}(\Sigma, \Lambda) \models_L \chi \text{ iff } \Sigma \cup \Lambda \models_L \chi.$$

Proof: Taking into account the equivalence $\phi \wedge \psi \rightarrow \psi' \equiv \phi \rightarrow (\psi \rightarrow \psi')$ we actually prove the lemma for $\mathrm{Red}(\Sigma, \Lambda) = \{\psi | (\phi \rightarrow \psi) \in \Sigma$ and the conjuncts of ϕ are in $\Lambda\}$, where ψ is written instead of $\psi \rightarrow \psi'$. The proof is by induction on the L-depth s of χ. The base case $s = 0$ is included in the inductive case $s \geq 1$ as the only difference is that $SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi) = SB_{\Sigma\cup\Lambda}(\chi) = \emptyset$ when $s = 0$.

($\Rightarrow$) Suppose $\Sigma \cup \Lambda \not\models_L \chi$, i.e. there exists a model $\mathcal{M}$ such that $\mathcal{M} \models \Sigma \cup \Lambda \cup SB_{\Sigma\cup\Lambda}(\chi)$ and $\mathcal{M} \not\models \chi$. By the definition of SB and the induction hypothesis $SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi) = SB_{\Sigma\cup\Lambda}(\chi)$. Let $(\phi \rightarrow \psi) \in \Sigma$. If $\psi \in \mathrm{Red}(\Sigma, \Lambda)$, then the conjuncts of ϕ are in Λ and $\mathcal{M} \models \phi$. Now $\mathcal{M} \models \psi$ because $\mathcal{M} \models \phi \rightarrow \psi$. This shows that $\mathcal{M} \models \mathrm{Red}(\Sigma, \Lambda) \cup SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi)$ and $\mathrm{Red}(\Sigma, \Lambda) \not\models_L \chi$.

($\Leftarrow$) Suppose $\mathrm{Red}(\Sigma, \Lambda) \not\models_L \chi$, i.e. there exists a model $\mathcal{M}$ such that $\mathcal{M} \models \mathrm{Red}(\Sigma, \Lambda) \cup SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi)$ and $\mathcal{M} \not\models \chi$. By the definition of SB and the induction hypothesis $SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi) = SB_{\Sigma\cup\Lambda}(\chi)$, and $\mathcal{M} \models SB_{\Sigma\cup\Lambda}(\chi)$. Let $\mathcal{M}'$ be $\mathcal{M}$ modified to satisfy Λ. This modification is possible without affecting the truth values of $\mathrm{Red}(\Sigma, \Lambda)$, $SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi)$, and χ because $\mathrm{Red}(\Sigma, \Lambda) \subseteq \mathcal{L}$, for $L\phi \in (Sf^{qL}(\Lambda) \cap Sf^{qL}(SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi)))$, $L\phi \in \Lambda$ iff $L\phi \in SB_{\mathrm{Red}(\Sigma,\Lambda)}(\chi)$ by the definition of SB and our assumption (similarly with $\neg L\phi$), and because the truth value of no propositional variable in χ changes and the $L\phi$ subformulae of χ stay unchanged because the sub-beliefs of χ also do.

We still have to show that $\mathcal{M}' \models \Sigma$. For $(\phi \rightarrow \psi) \in \Sigma$ if all conjuncts of ϕ are in Λ then $\psi \in \mathrm{Red}(\Sigma, \Lambda)$ and therefore $\mathcal{M} \models \psi$, and further, $\mathcal{M}' \models \phi \rightarrow \psi$. If at least one conjunct of ϕ is not in Λ then ϕ is false in $\mathcal{M}'$ (because then the complement of the conjunct is in Λ) and again $\mathcal{M}' \models \phi \rightarrow \psi$. Therefore $\Sigma \cup \Lambda \not\models_L \chi$. □

Lemma 4.4 *Let* ϕ *be of the form* $L\phi_1 \wedge \cdots \wedge L\phi_n \wedge \neg L\psi_1 \wedge \cdots \wedge \neg L\psi_m$ *and* $\Sigma \subseteq \mathcal{L}$. *Then* $\Sigma \models_L \phi$ *iff* $\Sigma \models_L \phi_i$ *for all* $i, 1 \leq i \leq n$ *and* $\Sigma \not\models_L \psi_j$ *for all* $j, 1 \leq j \leq m$.

Proof: ($\Rightarrow$) Suppose that for some i, $\Sigma \not\models_L \phi_i$, or for some j, $\Sigma \models_L \psi_j$. Then one of $L\phi_i$ or $\neg L\psi_j$ is not in $SB_\Sigma(\phi)$ and $\Sigma \not\models_L \phi$. ($\Leftarrow$) Suppose that the antecedent is true. Then $SB_\Sigma(\phi) = \{L\phi_1, \ldots, L\phi_n, \neg L\psi_1, \ldots, \neg L\psi_m\}$, and therefore $\Sigma \cup SB_\Sigma(\phi) \models \phi$ and $\Sigma \models_L \phi$. □

Let Λ be the Σ-full set corresponding to the unique stable expansion $SE_\Sigma(\Lambda)$ of a stratified set Σ. The following theorem gives an algorithm for computing directly the set $\mathrm{Red}(\Sigma, \Lambda) \subseteq \mathcal{L}$ which characterizes the stable expansion $SE_\Sigma(\Lambda)$.

Theorem 4.5 *Let* $\Sigma = \Sigma_1^n$ *be a stratified set. Let* $R = R_n$ *be defined by*

$$\begin{aligned} R_0 &= \emptyset \\ R_{i+1} &= R_i \cup \mathrm{Red}_L(\Sigma_{i+1}, R_i), 0 \leq i < n \end{aligned}$$

where $\mathrm{Red}_L(\Sigma, \Delta) = \{o(\phi) \rightarrow c(\phi) \mid \phi \in \Sigma, \Delta \models_L a(\phi)\}$. *Then* $R = \mathrm{Red}(\Sigma, \Lambda)$, *where* Λ *is the unique* Σ*-full set, and* $\{\phi \mid R \models_L \phi\}$ *is the unique stable expansion of* Σ.

Proof: We show by induction on i that for all $i = 0, \ldots, n$,

$$R_i = \mathrm{Red}(\Sigma_1^i, \Lambda_i). \tag{2}$$

For $i = 0$, $R_i = \mathrm{Red}(\Sigma_1^i, \Lambda_i) = \emptyset$. ($\subseteq$) By the induction hypothesis $R_{i-1} = \mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1})$ and therefore $R_{i-1} \subseteq \mathrm{Red}(\Sigma_1^i, \Lambda_i)$. Let $o(\phi) \rightarrow c(\phi) \in \mathrm{Red}_L(\Sigma_i, R_{i-1})$. Thus $R_{i-1} \models_L a(\phi)$. By the induction hypothesis $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \models_L a(\phi)$. The formula $a(\phi)$ is of the form $L\phi_1 \wedge \cdots \wedge L\phi_r \wedge \neg L\psi_1 \wedge \cdots \wedge \neg L\psi_s$ and by Lemma 4.4 for each $L\phi_j$, $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \models_L \phi_j$ and for each $\neg L\psi_j$, $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \not\models_L \psi_j$. By Lemma 4.3 $\Sigma_1^{i-1} \cup \Lambda_{i-1} \models_L \phi_j$ and $\Sigma_1^{i-1} \cup \Lambda_{i-1} \not\models_L \psi_j$. For each $L\phi_j$ there is some $k \leq i$ such that $L\phi_j \in Sf^L(\Sigma_k) - Sf^L(\Sigma_1^{k-1})$. By Lemma 4.2 $\Sigma_1^{k-1} \cup \Lambda_{k-1} \models_L \phi_j$ and thus $L\phi_j \in \Lambda_k \subseteq \Lambda_i$. Similarly it can be shown that $\neg L\psi_j \in \Lambda_i$. Thus $o(\phi) \rightarrow c(\phi) \in \mathrm{Red}(\Sigma_1^i, \Lambda_i)$ and $R_i \subseteq \mathrm{Red}(\Sigma_1^i, \Lambda_i)$.

($\supseteq$) Let $o(\phi) \rightarrow c(\phi) \in \mathrm{Red}(\Sigma_1^i, \Lambda_i)$. If $\phi \in \Sigma_1^{i-1}$, then $o(\phi) \rightarrow c(\phi) \in \mathrm{Red}(\Sigma_1^{i-1}, \Lambda_i) = \mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1})$. Therefore by the induction hypothesis $o(\phi) \rightarrow c(\phi) \in R_{i-1}$. Let $\phi \in \Sigma_i$. The conjuncts in $a(\phi)$ are in Λ_i where $a(\phi)$ is of the form $L\phi_1 \wedge \cdots \wedge L\phi_r \wedge \neg L\psi_1 \wedge \cdots \wedge \neg L\psi_s$. By Lemma 4.2 $\Sigma_1^{i-1} \cup \Lambda_{i-1} \models_L \phi_j$ and $\Sigma_1^{i-1} \cup \Lambda_{i-1} \not\models_L \psi_j$ for each $L\phi_j, \neg L\psi_j$. By Lemma 4.3 $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \models_L \phi_j$ and $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \not\models_L \psi_j$ and by the induction hypothesis $R_{i-1} \models_L a(\phi)$. Thus $o(\phi) \rightarrow c(\phi) \in \mathrm{Red}_L(\Sigma_i, R_{i-1}) \subseteq R_i$. Hence $\mathrm{Red}(\Sigma_1^i, \Lambda_i) \subseteq R_i$.

We have shown that $R_n = \mathrm{Red}(\Sigma_1^n, \Lambda_n)$. Thus $R_n \models_L \phi$ iff $\mathrm{Red}(\Sigma_1^n, \Lambda_n) \models_L \phi$ iff $\Sigma_1^n \cup \Lambda_n \models_L \phi$ by Lemma 4.3. Thus by Theorem 4.1 $\{\phi \mid R_n \models_L \phi\}$ is the unique stable expansion of Σ. □

Example 5 The table below demonstrates the computation of Λ by the algorithm in Theorem 4.1, and the computation of $R = \mathrm{Red}(\Sigma, \Lambda)$ with Theorem 4.5.

i	Σ_i	Λ	R
2	$L(p \leftrightarrow q) \rightarrow s$	$L(p \leftrightarrow q)$	s
1	$L\neg Lr \wedge p \rightarrow q$ $q \rightarrow p$	$\neg Lr, L\neg Lr$	$p \rightarrow q$ $q \rightarrow p$

■

5 COMPLEXITY RESULTS

We write $|\phi|$ for the length of a formula ϕ, $|\Sigma|$ for the sum of lengths of the formulae in a set Σ, and $\|\Sigma\|$ for the cardinality of a set Σ.

As a basis of analyzing the complexity of computing the stable expansions of stratified sets we use the algorithm in Theorem 4.5 for computing a set $R \subseteq \mathcal{L}$ that characterizes the unique stable expansion of a stratified set. By the next lemma the explicit construction of the sets $SB_\Sigma(\chi)$ can be avoided in the $\models_L$ tests of the algorithm. As a result all consequence tests are for sets of formulae in which no L operators appear.

Lemma 5.1 *Let $\Sigma \subseteq \mathcal{L}$ and $\chi(L\phi)$ be a formula in which the formula $L\phi$ possibly occurs and $\chi(\top)$ the same formula where all occurrences of $L\phi$ have been replaced by the constant true $\top$. Now $\Sigma \cup \{L\phi\} \models \chi(L\phi)$ iff $\Sigma \models \chi(\top)$, and $\Sigma \cup \{\neg L\phi\} \models \chi(L\phi)$ iff $\Sigma \models \chi(\bot)$, where $\bot = \neg\top$.*

Proof: ($\Leftarrow$) Suppose $\Sigma \cup \{L\phi\} \not\models \chi(L\phi)$, i.e. there is a model M such that $\mathcal{M} \models \Sigma \cup \{L\phi\}$ and $\mathcal{M} \not\models \chi(L\phi)$. Clearly $\mathcal{M} \not\models \chi(\top)$. ($\Rightarrow$) Suppose $\Sigma \not\models \chi(\top)$, i.e. $\mathcal{M} \models \Sigma$ and $\mathcal{M} \not\models \chi(\top)$. Since the truth value of $L\phi$ is independent both of other formulae of the form $L\psi$ and of propositional variables, $\top$ can be replaced by $L\phi$ in χ and therefore $\Sigma \cup \{L\phi\} \not\models \chi(L\phi)$. Similarly for $\neg L\phi$ and $\bot$. □

Lemma 5.2 *$F(\chi, \Sigma)$ in Equation 3 gives the amount of resources needed for testing the $\models_L$ consequence of an arbitrary formula χ from a set of formulae $\Sigma \subseteq \mathcal{L}$. $R(x)$ is the amount of resources needed for performing a consequence test of size x.*

$$F(\chi,\Sigma) = R(|\chi| + |\Sigma| - \sum_{L\phi \in Sf^{qL}(\chi)} |\phi|) + \sum_{L\phi \in Sf^{qL}(\chi)} F(\phi,\Sigma) \qquad (3)$$

and the equation with recursion removed is

$$F(\chi,\Sigma) = R(|\chi| + |\Sigma| - \sum_{L\phi \in Sf^{qL}(\chi)} |\phi|) + \sum_{L\phi \in Sf^{L}(\chi)} R(|\phi| + |\Sigma| - \sum_{L\psi \in Sf^{qL}(\phi)} |\psi|) \qquad (4)$$

Proof: The second summand of the rhs of Equation 3 corresponds to the computation of the members of $SB_\Sigma(\chi)$ using $\models_L$. The first summand corresponds to the consequence test $\Sigma \models_L \chi$, i.e. the test of satisfiability of $\{\neg\chi\} \cup \Sigma \cup SB_\Sigma(\chi)$. By Lemma 5.1 the consequence test can be made by replacing $L\phi$ in χ by $\top$ for all $L\phi \in SB_\Sigma(\chi)$, and by $\bot$ for all $\neg L\phi \in SB_\Sigma(\chi)$, thus obtaining $\chi' \in \mathcal{L}$ and then testing the consequence $\Sigma \models \chi'$. This is why $\sum_{L\phi \in Sf^{qL}(\chi)} |\phi|$ is subtracted. □

Next we give an upper bound for the amount of resources needed for consequence tests in the computation of the algorithm in Theorem 4.5. Let $\Sigma = \Sigma_1^n$ be a stratified set and let there be an enumeration of the formulae $\phi_i \in \Sigma, 1 \leq i \leq r$ such that if the stratum of ϕ_i is lower than that of ϕ_j then $i < j$. Let $n_i = |a(\phi_i)|$ and $m_i = |o(\phi_i) \rightarrow c(\phi_i)|$. Ignoring the exact boundaries between the strata is an acceptable approximation since we are primarily interested in analyzing the upper bounds of complexity. The size of a stratified set is the sum of the sizes of its formulae:

$$\sum_{i=1}^{r} (n_i + m_i + 1) \qquad (5)$$

and an upper bound for the resources needed for the consequence tests is

$$\sum_{i=1}^{r} F(a(\phi_i), \bigcup_{j=1}^{i-1} \{o(\phi_j) \rightarrow c(\phi_j)\}) \qquad (6)$$

Tractable classes of stratified sets of autoepistemic formulae can be found by restricting the syntactic form of the formulae in such a way that the classical theorem proving task becomes tractable. As an example we present a tractable class SHC_{ae} based on Horn clauses, i.e. disjunctions of literals of which at most one is positive. For the members of this class the $\models$ consequence can be tested by using the linear time algorithm of Dowling and Gallier [1984] and for SHC_{ae} the algorithm of the previous section runs in polynomial time.

Definition 5.3 *A formula χ is in the class HF_{ae} if it is a disjunction of conjunctions of formulae of the form p, $\neg p$, $L\phi$, and $\neg L\phi$ with at most one $\neg p$ in each disjunct, where each p is a propositional variable and each ϕ is in HF_{ae}.*

Definition 5.4 *SHC_{ae} is the class of finite stratified sets of formulae ϕ of the form $a(\phi) \wedge o(\phi) \rightarrow c(\phi)$ where $a(\phi)$ is a conjunction of zero or more formulae of the form $L\chi$ or $\neg L\chi$ where χ is in HF_{ae}, $o(\phi)$ is a conjunction of zero or more propositional variables, and $c(\phi)$ is a propositional variable or a negated propositional variable.*

The following are examples of the syntactic form of the formulae in sets in SHC_{ae}.

$$a \wedge b \rightarrow c$$
$$\neg L(LLp \vee q \vee \neg r \vee (t \wedge \neg u) \vee (x \wedge y)) \rightarrow \neg a$$

Theorem 5.5 *For a set Σ in SHC_{ae} the set $\mathrm{Red}(\Sigma, \Lambda)$, where Λ is the unique Σ-full set, can be computed in time $\mathcal{O}(n^2)$ where $n = |\Sigma|$.*

Proof: By Proposition 3.5 a stratification can be determined in $\mathcal{O}(|\Sigma|)$ time, and the rest of the computation of the algorithm in Theorem 4.5 is clearly dominated by the consequence tests. All consequence tests reduce to satisfiability tests for formulae that are in conjunctive normal form with at most one positive literal in each conjunct, so that the linear time satisfiability algorithm of [Dowling and Gallier, 1984] can be used. For logical consequence tests in the computation of the algorithm in Theorem 4.5, instantiate $R(x) = ax + c$ to the second equation of Lemma 5.2:

$$F(\chi, \Sigma) = a(|\chi| + |\Sigma| - \sum_{L\phi \in Sf^{qL}(\chi)} |\phi|) +$$
$$c + \sum_{L\phi \in Sf^{L}(\chi)} (a(|\phi| + |\Sigma| - \sum_{L\psi \in Sf^{qL}(\phi)} |\psi|) + c)$$

By reordering the terms:

$$F(\chi, \Sigma) = a(|\chi| + |\Sigma| + \sum_{L\phi \in Sf^{L}(\chi)} |\Sigma|) +$$
$$(c + \sum_{L\phi \in Sf^{L}(\chi)} c) + a(- \sum_{L\phi \in Sf^{qL}(\chi)} |\phi| + \sum_{L\phi \in Sf^{L}(\chi)} |\phi|$$
$$- \sum_{L\phi \in Sf^{L}(\chi)} \sum_{L\psi \in Sf^{qL}(\phi)} |\psi|)$$

The value of the third summand is zero since the length of each $L\phi \in Sf^L(\chi)$ is added and subtracted exactly once. From this we get the upper bound

$$\begin{aligned} &a(|\chi| + |\Sigma| + \|Sf^L(\chi)\| \cdot |\Sigma|) + c(1 + \|Sf^L(\chi)\|) \\ &\leq a(|\chi| + |\Sigma| \cdot (1 + \|Sf^L(\chi)\|)) + c \cdot |\chi|) \\ &\leq a(|\chi| + |\Sigma| \cdot |\chi|) + c \cdot |\chi| \\ &< (a + c)(|\chi| + |\Sigma| \cdot |\chi|) \end{aligned}$$

for $F(\chi, \Sigma)$. Using this and Equation 6 (by the definition of $\mathcal{O}$ the constant factor $a + c$ appearing in each term can be left out).

$$\begin{aligned} &\sum_{i=1}^{r}(n_i + n_i \sum_{j=1}^{i-1} m_i) \leq (\sum_{i=1}^{r} n_i) + (\sum_{i=1}^{r} n_i)(\sum_{i=1}^{r} m_i) \\ &\leq (\sum_{i=1}^{r} n_i)^2 + 2(\sum_{i=1}^{r} n_i)(\sum_{i=1}^{r} m_i) + (\sum_{i=1}^{r} m_i)^2 \\ &< (\sum_{i=1}^{r} n_i + m_i + 1)^2 \end{aligned}$$

Thus, we obtain the $\mathcal{O}(n^2)$ upper bound for the logical consequence tests. □

Theorem 5.6 *Given $\mathrm{Red}(\Sigma, \Lambda)$ where $\Sigma \in SHC_{ae}$ and Λ is a Σ-full set, the membership in the unique stable expansion of Σ $\{\phi | \mathrm{Red}(\Sigma, \Lambda) \models_L \phi\}$ for formulae ϕ in HF_{ae} can be decided in time $\mathcal{O}(n^2)$, where $n = |\Sigma| + |\neg\phi|$.*

Proof: By the second equation of Lemma 5.2 the amount of resources needed is bounded by $|\Sigma| \cdot |\neg\phi| + |\neg\phi|$ as shown in the proof of Theorem 5.5, and this is $\mathcal{O}(n^2)$. □

Theorem 5.7 *Let Σ be in SHC_{ae}. The membership problem of the unique stable expansion of Σ for formulae ϕ in HF_{ae} is solvable in time $\mathcal{O}(n^2)$ where $n = |\Sigma| + |\neg\phi|$.*

The line taken in establishing the above result suggests further tractable classes based on other subsets of propositional logic for which polynomial time satisfiability tests are available, like those presented in [Schaefer, 1978; Gallo and Scutella, 1988]. Next we investigate an even more restricted class for which the logical consequence testing does not have to be done separately for each formula inside L.

Definition 5.8 *A formula χ is in CF_{ae} if it is a conjunction of one or more formulae of the form p, $L\phi$, and $\neg L\phi$ where each p is a propositional variable and each ϕ is in CF_{ae}.*

Definition 5.9 *SPC_{ae} is the class of finite stratified sets of formulae ϕ of the form $a(\phi) \wedge o(\phi) \rightarrow c(\phi)$ where $a(\phi)$ is a conjunction of zero or more formulae of the form $L\chi$ or $\neg L\chi$ and each χ is in CF_{ae}, $o(\phi)$ is a conjunction of zero or more propositional variables, and $c(\phi)$ is a propositional variable.*

The following formulae illustrate the form of formulae in sets in SPC_{ae}:

$$\neg L(a \wedge b \wedge c \wedge \neg Ld) \rightarrow e$$
$$L(p \wedge Lq \wedge \neg Lr) \wedge \neg L(\neg Ls) \wedge t \rightarrow u$$

In [Dowling and Gallier, 1984] two linear time algorithms are given for testing the satisfiability of a set of Horn clauses. We modify the first one of these to be used in linear time tests for the membership in the unique stable expansions of sets in SPC_{ae}. By Theorem 4.5 explicit construction of the sets Λ_i can be avoided and instead of separately testing the logical consequence of each formula inside L in $a(\phi)$ the whole set of consequences for one stratum of a stratification can be computed by one run of the algorithm.

The function DG is the basis of the efficient algorithm for SPC_{ae} and can be implemented as a variant of Algorithm 2 of [Dowling and Gallier, 1984]. Dowling and Gallier's algorithm works with arbitrary Horn clauses, but ours is restricted to *program clauses*, i.e. disjunctions of literals exactly one of which is positive. Sometimes program clauses are written as $a_1 \wedge \cdots \wedge a_n \rightarrow b$ instead of $\neg a_1 \vee \cdots \vee \neg a_n \vee b$.

In our algorithm sets of propositional variables are represented by their characteristic functions or equivalently arrays. For array indexing the Assumption 3.4 is essential.

Proposition 5.10 *Let Σ be a set of program clauses and v a set of propositional variables. Then*

$DG(\Sigma, v) = \{p | p$ *is a propositional variable,* $\Sigma \cup v \models p\}$

can be computed in time $\mathcal{O}(|\Sigma|)$.

Proof: The computation of *DG* for a set of program clauses Σ and a set of propositional variables v uses the following arrays:

poslitlist Each element n is initialized to the propositional variable appearing in the positive literal of the clause $\phi_n \in \Sigma$. The initialization can be done by one traversal of the set of formulae, which is linear time.

clauselist The element n is the list of clauses of Σ in which the variable in the positive literal of ϕ_n appears in a negative literal. Initialization can be done in linear time.

numargs The element n is initialized to the number of variables p in the negative literals of ϕ_n for which $v(p) = 0$.

The function is computed by the following procedure.

```
function DG(Σ, v:array of {0,1}) : array of {0,1};
begin
  initialize poslitlist, clauselist, numargs;
  initialize queue to the list of clauses n
  for which numargs[n] = 0;
  for each c in queue do v(poslitlist[c]) := 1;
  while queue ≠ empty do
   clause1 := pop(queue);
   for each clause2 in clauselist[clause1] do
    numargs[clause2] := numargs[clause2] - 1;
    if numargs[clause2] = 0 then
     n := poslitlist[clause2];
     if v(n) = 0 then
      v(n) := 1;
      queue := push(clause2,queue);
     end if
    end if
   end for
  end while
  return v;
end
```

Under Assumption 3.4 the computation is linear time on the size of Σ. The amount of computation inside the while loop is proportional to the number of negative literals in Σ. □

Computing $\mathrm{Red}_L(\Sigma, v) = \{\phi \rightarrow \phi' | (L\chi_i \wedge \cdots \wedge L\chi_n \wedge \neg L\chi_{n+1} \wedge \cdots \wedge \neg L\chi_{n+m} \wedge \phi \rightarrow \phi') \in \Sigma, v \models_L \chi_1, \ldots, v \models_L \chi_n, v \not\models_L \chi_{n+1}, \ldots, v \not\models_L \chi_{n+m}\}$ can be done in linear time on $|\Sigma|$ for members of SPC_{ae}. The consequence tests $v \models_L \chi$, χ in CF_{ae} can be done in linear time on $|\chi|$ by using the following algorithm.

$v \models_L \chi$ iff

$$\text{for each conjunct } \phi \text{ of } \chi \begin{cases} \text{if } \phi \text{ is atomic then } \phi \in v \\ \text{if } \phi = L\psi \text{ then } v \models_L \psi \\ \text{if } \phi = \neg L\psi \text{ then } v \not\models_L \psi \end{cases}$$

The following two lemmata are needed for establishing Lemma 5.13 which describes how to compute in $\mathcal{O}(|\Sigma|)$ time the set of propositional variables in the unique stable expansion of a set Σ in SPC_{ae}.

Lemma 5.11 *Let χ be in CF_{ae}, $\Sigma \subseteq \mathcal{L}$, and $v = \{p | p$ is a propositional variable, $\Sigma \models p\}$. Then $v \models_L \chi$ iff $\Sigma \models_L \chi$.*

Proof: By induction on the L-depth s of χ. Let $s = 0$. Suppose $\Sigma \not\models_L \chi$, i.e. there is a model $\mathcal{M}$ for which $\mathcal{M} \models \Sigma$ and $\mathcal{M} \not\models \chi$. Because v is the set of propositional variables true in every model of Σ, also $\mathcal{M} \models v$, and $v \not\models_L \chi$. Suppose $\Sigma \models_L \chi$. Because χ is a conjunction of propositional variables each of which is a logical consequence of Σ, obviously $v \models_L \chi$. Let $s \geq 1$. By the induction hypothesis $SB_v(\chi) = SB_\Sigma(\chi)$, and replacing members of $SB_v(\chi)$ (and $SB_\Sigma(\chi)$) according to Lemma 5.1 by $\top$ or $\bot$, χ can be reduced to χ' of L-depth 0 for which $v \models \chi'$ iff $\Sigma \models \chi'$, which can be shown as in the case $s = 0$. □

Lemma 5.12 *Let Σ be a set of program clauses and $\Gamma = \{p | p$ is a propositional variable, $\Sigma \models p\}$. Let Δ be a set of program clauses such that the propositional variables in the positive literals of Δ do not occur in the negative literals of Σ. Then for all propositional variables p, $\Sigma \cup \Delta \models p$ iff $\Gamma \cup \Delta \models p$.*

Proof: Suppose $\Sigma \cup \Delta \not\models p$, i.e. for some model $\mathcal{M}$, $\mathcal{M} \models \Sigma \cup \Delta$ and $\mathcal{M} \not\models p$. Clearly $\mathcal{M} \models \Gamma \cup \Delta$ and therefore $\Gamma \cup \Delta \not\models p$. Suppose $\Gamma \cup \Delta \not\models p$, i.e. there is a model $\mathcal{M}$, $\mathcal{M} \models \Gamma \cup \Delta$, $\mathcal{M} \not\models p$. $\mathcal{M}'$ is $\mathcal{M}$ modified to satisfy $\Sigma \cup \Delta$ in the following way. Formulae $\phi = \neg a_1 \vee \cdots \vee \neg a_n \vee b, \phi \in \Sigma$ can be false if $\mathcal{M} \models a_1 \wedge \cdots \wedge a_n$ and $\mathcal{M} \not\models b$. Now $\mathcal{M}'$ can be modified to make ϕ true by making one of $a_i, 1 \leq i \leq n$ false. This can be done without falsifying formulae in $\Gamma \cup \Delta$ because i) not all $a_i, 1 \leq i \leq n$ can be logical consequences of Σ. If they were, then $b \in \Gamma$. And ii) as propositional variables in the positive literals of Δ do not occur in the negative

literals of Σ, this modification falsifies no formula in Δ. Therefore $\mathcal{M}' \models \Sigma \cup \Delta$ and $\Sigma \cup \Delta \not\models p$. □

Lemma 5.13 *For $\Sigma = \Sigma_1^n$ in SPC_{ae} and formulae χ in CF_{ae}, $v_n \models_L \chi$ iff χ belongs to the unique stable expansion of Σ, where v_n is defined by*

$$\begin{aligned} v_0 &= \emptyset \\ v_{i+1} &= DG(\mathrm{Red}_L(\Sigma_{i+1}, v_i), v_i), 0 \leq i < n. \end{aligned}$$

Furthermore, v_n, which is the set of propositional variables in the unique stable expansion of Σ, can be computed in time $\mathcal{O}(|\Sigma|)$.

Proof: We prove $v_i \models_L \chi$ iff $\mathrm{Red}(\Sigma_1^i, \Lambda_i) \models_L \chi$, where Λ_i as in Theorem 4.1, by induction on i. Let $i = 0$. Immediate as $v_0 = \mathrm{Red}(\Sigma_1^0, \Lambda_0) = \emptyset$. Let $i > 0$. For all propositional variables p,

$$\begin{aligned} v_i \models p \quad &\text{iff} \quad v_{i-1} \cup \mathrm{Red}_L(\Sigma_i, v_{i-1}) \models p \\ &\text{iff} \quad \mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \cup \mathrm{Red}_L(\Sigma_i, v_{i-1}) \models p \\ &\text{iff} \quad \mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \cup \mathrm{Red}(\Sigma_i, \Lambda_i) \models p \\ &\text{iff} \quad \mathrm{Red}(\Sigma_1^i, \Lambda_i) \models p. \end{aligned}$$

The first equivalence is by the definition of v_i, the second and the third by **A** and **B** below, respectively. **A.** By the induction hypothesis $v_{i-1} \models p$ iff $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \models p$. We get the equivalence by Lemma 5.12 taking $\Delta = \mathrm{Red}_L(\Sigma_i, v_{i-1}), \Gamma = v_{i-1}, \Sigma = \mathrm{Red}(\Sigma_1^{i-1}, \Lambda_i)$. **B.** By the definitions of Red and Red_L a formula ϕ is in $\mathrm{Red}_L(\Sigma_i, v_{i-1})$ if $(\psi \rightarrow \phi) \in \Sigma_i$ and for the conjuncts $L\chi, \neg L\chi'$ of ψ, $v_{i-1} \models_L \chi$ and $v_{i-1} \not\models_L \chi'$, and in $\mathrm{Red}(\Sigma_i, \Lambda_i)$ if $L\chi, \neg L\chi' \in \Lambda_i$. By Lemmata 4.2 and 4.3 $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \models_L \chi$ and $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \not\models_L \chi'$. By the induction hypothesis $v_{i-1} \models_L \chi$ iff $\mathrm{Red}(\Sigma_1^{i-1}, \Lambda_{i-1}) \models_L \chi$. Therefore $\mathrm{Red}_L(\Sigma_i, v_{i-1}) = \mathrm{Red}(\Sigma_i, \Lambda_i)$.

By Lemma 5.11 we get the induction step, i.e. for all χ in CF_{ae}, $v_i \models \chi$ iff $\mathrm{Red}(\Sigma_1^i, \Lambda_i) \models \chi$.

By Theorem 2.4 and Lemma 4.3 $\chi \in \mathrm{CF}_{ae}$ is in the unique stable expansion of Σ iff $v_i \models_L \chi$.

By Proposition 3.5 stratification can be computed in linear time, and using Assumption 3.4 the computation is $\mathcal{O}(|\Sigma|)$ time because computing $\mathrm{Red}_L(\Sigma_i, v_{i-1})$ is linear in $|\Sigma_i|$ and this is done exactly once for each $\Sigma_i \subseteq \Sigma, 1 \leq i \leq n$. The size of $\mathrm{Red}_L(\Sigma_i, v_{i-1})$ is smaller than or equal to that of Σ_i and therefore the respective computation with DG is linear in $|\Sigma_i|$. □

Theorem 5.14 *Let Σ be in SPC_{ae} and χ in CF_{ae}. The membership of χ in the unique stable expansion of Σ can be decided in time $\mathcal{O}(n)$, where $n = |\Sigma| + |\chi|$.*

Proof: By Lemma 5.13 the computation of the set v_n for $\Sigma = \Sigma_1^n$ is $\mathcal{O}(|\Sigma|)$. The membership in the unique stable expansion can be tested by $v_n \models_L \chi$, and this is $\mathcal{O}(|\chi|)$. □

6 APPLICATIONS

Autoepistemic logic is closely related to McDermott and Doyle style nonmonotonic modal logics. Marek et al. [1991] show that for a wide range of modal logics S, S-expansions coincide with stable expansions in the stratified case. Thus all the methods and results obtained for stratified autoepistemic logic are directly applicable to these logics.

Theorem 6.1 (Marek et al. [1991]) *Let $\Sigma \subseteq \mathcal{L}_{ae}$ be stratified. Then for each logic S such that $\mathbf{N} \subseteq S$ and $S \subseteq \mathbf{KD45}$ or $S \subseteq \mathbf{S4}$ the unique stable expansion of Σ is the unique S-expansion of Σ.*

Reiter's [1980] default logic is one of the leading formalizations of nonmonotonic reasoning. Konolige [1988] shows that under a suitable translation of a default theory extensions of the theory correspond to stable expansions of the translated theory satisfying a special groundedness condition. In the case of stratified default theories the special groundedness condition is not required. So the iterative algorithm in Theorem 4.5 yields efficient methods for computing extensions of stratified default theories based on subsets of propositional logic for which efficient satisfiability tests are available. As a straightforward example of this we show that the linear time algorithm in Lemma 5.13 implies a linear time algorithm for a similar class of stratified default logic.

We call (D, W) a stratified Horn default theory if

1. W is a finite set of propositional program clauses and
2. D is a finite set of default rules $\frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma}$ where the prerequisite α is a conjunction of propositional variables and each justification β_i is a disjunction of negated propositional variables and the conclusion γ is a propositional variable and there is a partition $D_1, \ldots, D_n$ of D such that if γ is a conclusion of a default rule in D_i, then γ does not appear in $D_1 \cup \ldots \cup D_{i-1}$, in negative literals in W nor in any prerequisite or justification in D_i.

Lemma 6.2 *Let (D, W) be a stratified Horn default theory. Then $E = \Delta \cap \mathcal{L}$ is the unique extension of (D, W) where Δ is the unique stable expansion of the stratified autoepistemic theory*

$$W \cup \{L\alpha \wedge \neg L\neg\beta_1 \wedge \ldots \wedge \neg L\neg\beta_n \rightarrow \gamma \mid \frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma} \in D\}. \quad (7)$$

Theorem 6.3 *Let (D, W) be a stratified Horn default theory. Then the atomic part of the unique extension of (D, W) can be computed in linear time.*

Stable model semantics [Gelfond and Lifschitz, 1988] and well-founded semantics [Van Gelder *et al.*, 1991] are the leading declarative semantics for general logic programs. For stratified propositional logic programs the stable model semantics and the well-founded semantics coincide. Logic programs can be seen as sets of autoepistemic formulae, e.g. using the following translation proposed by Gelfond and Lifschitz [1988].

$$\mathrm{tr}_{LP}(a_0 \leftarrow a_1, \ldots, a_m, \mathrm{not}\, a_{m+1}, \ldots, \mathrm{not}\, a_n) = (a_1 \wedge \ldots \wedge a_m \wedge \neg L a_{m+1} \wedge \ldots \wedge \neg L a_n) \rightarrow a_0 \quad (8)$$

When using this translation stable models and stable expansions are closely related in the stratified case. The correspondence is described in the following lemma which a direct consequence of the results of Gelfond and Lifschitz [1988].

Lemma 6.4 *Let P be a propositional stratified general logic program. $S = \Delta \cap \mathrm{At}$ is the unique stable model (well-founded model) of P where Δ is the unique stable expansion of $\mathrm{tr}_{LP}(P)$ and* At *is the set of propositional variables.*

Deciding whether a propositional logic program has a stable model is NP-complete in the general case [Marek and Truszczyński, 1991]. Kautz and Selman [1991] propose a cubic algorithm for computing the stable model of a stratified logic program using a translation to default logic. This result can be improved using the algorithm in Lemma 5.13.

Theorem 6.5 *Let P be a stratified general propositional logic program. The unique stable model (well-founded model) of P can be computed in linear time.*

Elkan [1990] shows that the justifications in a nonmonotonic truth maintenance system (TMS) can be seen as a propositional general logic program and that the grounded model computed by the TMS is a stable model of the corresponding logic program.

Corollary 6.6 *Let J be a stratified set of justifications. Then the unique grounded model of J can be computed in linear time.*

7 CONCLUSIONS

The attempts to find subclasses of nonmonotonic reasoning which can be implemented efficiently by limiting the computational complexity of required classical reasoning have produced very disappointing results [Kautz and Selman, 1991; Elkan, 1990; Marek and Truszczyński, 1991]. In this paper we follow a different approach. We identify the ability to "define" propositions using default assumptions about the same propositions as a source of additional computational complexity in nonmonotonic reasoning. Disallowing such constructs, i.e. requiring the knowledge base to be stratified, gives a significant computational advantage. We demonstrate this by developing an iterative (non-backtracking) algorithm for stratified autoepistemic theories the complexity of which is dominated by required classical reasoning. Thus efficient subclasses of stratified nonmonotonic reasoning can be obtained by restricting the form of sentences in the knowledge base. As an example, we develop a quadratic and a linear time algorithm for limited subclasses of stratified autoepistemic theories. The results are shown to imply fast reasoning methods for stratified cases of default logic, logic programs, truth maintenance systems and nonmonotonic modal logics. E.g., deciding whether a propositional logic program has a stable model is an NP-complete problem in the general case but for the stratified case we give a linear time algorithm which computes the unique stable model.

Acknowledgements

Ilkka Niemelä gratefully acknowledges the support from the following foundations: Foundation of Technology, Jenny and Antti Wihuri Foundation, Emil Aaltonen Foundation, Alfred Kordelin Foundation, as well as Heikki and Hilma Honkanen Foundation.

References

A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley, 1974.

K.R. Apt, H.A. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann Publishers, Los Altos, 1988.

A. K. Chandra and D. Harel. Horn clause queries and generalizations. *Journal of Logic Programming*, 2(1):1–15, 1985.

W.F. Dowling and J.H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, (3):267–284, 1984.

C. Elkan. A rational reconstruction of nonmonotonic truth maintenance systems. *Artificial Intelligence*, 43:219–234, 1990.

G. Gallo and M. G. Scutella. Polynomially solvable satisfiability problems. *Information Processing Letters*, 29:221–226, November 1988.

M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference on Logic Programming*,

pages 1070–1080, Seattle, USA, August 1988. MIT Press.

M. Gelfond. Autoepistemic logic and formalization of commonsense reasoning, preliminary report. In *Proceedings of the 2nd Workshop on Non-Monotonic Reasoning*, pages 176–186, Grassau, FRG, June 1988. Springer-Verlag.

G. Gottlob. Complexity results for nonmonotonic logics. Technical Report CD-TR 91/24, Christian Doppler Laboratory for Expert Systems, Institut für Informationssysteme, Technische Universität Wien, Vienna, Austria, August 1991.

H. Kautz and B. Selman. Hard problems for simple default theories. *Artificial Intelligence*, 49:243–279, 1991.

Phokion G. Kolaitis. The expressive power of stratified programs. *Information and Computation*, 90(1):50–66, January 1991.

K. Konolige. On the relation between default and autoepistemic logic. *Artificial Intelligence*, 35:343–382, 1988.

C. Lassez, K. McAloon, and G. Port. Stratification and knowledge base management. In *Proceedings of the 4th International Conference on Logic Programming*, volume 1, pages 136–151, 1987.

W. Marek and M. Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38:588–619, 1991.

W. Marek, G.F. Shvarts, and M. Truszczyński. Modal nonmonotonic logics: Ranges, characterization, computation. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 395–404, Cambridge, MA, USA, April 1991. Morgan Kaufmann Publishers.

R.C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94, 1985.

I. Niemelä. Towards automatic autoepistemic reasoning. In *Proceedings of the European Workshop on Logics in Artificial Intelligence—JELIA '90*, pages 428–443, Amsterdam, The Netherlands, September 1990. Springer-Verlag.

R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.

R. E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal of Computing*, 1(2):146–160, 1972.

A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, July 1991.

A. Van Gelder. Negation as failure using tight derivations for general logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 149–176. Morgan Kaufmann Publishers, Los Altos, 1988.

All You Ever Wanted to Know about Tweety (But Were Afraid to Ask)

Gerhard Lakemeyer
Institut für Informatik III
Universität Bonn
Römerstraße 164
W-5300 Bonn 1, Germany
gerhard@uran.informatik.uni-bonn.de

Abstract

We propose a formalization of only-knowing-about, which captures the idea that something is all an agent knows about some subject matter. The work extends a previous formalization of only-knowing by Levesque. Besides discussing some of the logical properties of the new concept, we also address the issue of computing what is known about some subject matter for a given knowledge base. In this context, we are able to relate only-knowing-about to deKleer's ATMS. Finally, we show that only-knowing-about is efficiently computable, if we weaken the underlying model of belief.

1 All I know about...

In [Lev90], Levesque proposes a formalization of the concept of *only-knowing*. This notion, which captures the idea that something is *all* an agent knows,[1] has proven very valuable in characterizing the knowledge of a fully introspective agent. In particular, Levesque uses it to reconstruct and in fact generalize Moore's autoepistemic logic within the framework of a classical (monotonic) logic.

While a logic of only-knowing is thus very useful from a theoretical point of view, being able to talk about *all* an agent knows does not seem very relevant from a practical point of view, say, as part of a query language. After all, hardly anybody is interested in all I know.[2] A much more useful concept in this regard seems to be the notion of *only-knowing-about*, where one is interested in all the agent knows about a certain subject matter. Thus queries like *is this all you know about Tweety* or *what do you know about Tweety* seem very appropriate.

Another application is a multi-agent scenario where agents reason about other agents, possibly applying defaults. For example, assume there are two agents, Jack and Jill, with the default that birds fly unless known otherwise being common knowledge. If Jill knows that all Jack knows about Tweety is that he is a bird, then Jill knows that Jack knows that Tweety flies. Notice that, while it is unreasonable to assume that an agent knows all another agent knows, it is quite possible that an agent knows all another agent knows about a certain subject. For example, Jill may just have bought Tweety and now tells Jack that Tweety is a bird.

In this paper, we propose a formalization of *only-knowing-about* in the propositional case. We do so by extending the existing logic of *only-knowing* developed by Levesque [Lev90]. Besides discussing general properties of only-knowing-about, we also address the issue of computing all an agent knows about a subject matter and relate it to deKleer's ATMS [deK86]. Finally, we show how, by weakening the underlying model of belief, this computation can be carried out efficiently.

To our knowledge, this is the first formalization of only-knowing-about that has been studied in detail. Levesque, in his paper on only-knowing, also suggests a definition of only-knowing-about which unfortunately has serious deficiencies. We will get back to this later. Preliminary results of our work appears in [Lak92].

The paper is organized as follows. In Section 2, we introduce Levesque's logic of only-knowing. In Section 3, we extend Levesque's logic by introducing the concept of only-knowing-about and discuss some of its properties. Section 4 addresses the issue of computing all that is known about some subject matter for a given knowledge base. Section 5 shows how all that is known about some subject matter can be computed efficiently if we weaken the underlying model of belief appropriately. Finally, Section 6 summarizes the

[1] We will freely use the terms *knowledge* and *belief* interchangeably. While all of the logics discussed here allow an agent to have false beliefs, none of our results hinges on this choice.

[2] Besides, for a knowledge-based agent it is a trivial matter to tell a user all it knows. It only needs to hand over a copy of its knowledge base.

results and discusses future directions.

2 The Logic *OL*

In this section, we introduce a propositional version of Levesque's logic of only-knowing.[3]

The primitives of the language are a countably infinite set $\mathcal{P}$ of atomic propositions (or atoms), the connectives $\vee$, $\neg$, and the modal operators **L** and **O**. For convenience we also include a special atom $\Box$, which always denotes falsity. Sentences are formed in the usual way from these primitives.[4]

Literals are either atoms or negated atoms. **Clauses** are disjunctions of literals. A clause c is contained in a clause c' ($c \subseteq c'$) if every literal in c other than $\Box$ occurs in c'. We write $c \subsetneq c'$ instead of $c \subseteq c'$ and $c' \not\subseteq c$. A sentence is called **objective** if it contains no modal operator, **subjective** if every atom occurs within the scope of a modal operator, and **basic** if it contains no **O**'s.

The semantics of *OL* is based on the notion of a world, which is simply a truth assignment of the atoms.

Definition 1 (Worlds)
A world w is a function $w : \mathcal{P} \longrightarrow \{\mathbf{t}, \mathbf{f}\}$ such that $w(\Box) = \mathbf{f}$.

Belief (**L**) is interpreted possible-world style.[5] The idea is that an agent imagines a set of worlds all of which are compatible with what the agent believes about the world. In other words, the agent believes a sentence just in case that sentence is true in all the worlds he or she imagines.

In order for $\mathbf{O}\alpha$ to capture the intuition that α is all that is believed, we require that α not only holds at all the worlds that are imagined, i.e., that α is believed, but also that the set of worlds is as large as possible. In other words, we could not possibly add more worlds to the set and still believe α. If α is objective, there is a unique M that has this property, namely the set of *all* worlds where α is true. If α itself contains modal operators, there may be multiple M or none at all, a phenomenon that corresponds precisely to the fact that a sentence may have multiple stable expansions or none at all [Lev90].

Let p be an atom and let α and β be arbitrary sentences, The truth of a sentence α with respect to a world w and a set of worlds M ($M, w \models \alpha$) is defined as follows:

$$
\begin{array}{lcl}
M, w \models p & \iff & w(p) = \mathbf{t} \\
M, w \models \neg\alpha & \iff & M, w \not\models \alpha \\
M, w \models \alpha \vee \beta & \iff & M, w \models \alpha \text{ or } M, w \models \beta \\
M, w \models \mathbf{L}\alpha & \iff & \text{for all } w' \in M,\ M, w' \models \alpha \\
M, w \models \mathbf{O}\alpha & \iff & \text{for all } w',\ w' \in M \text{ iff } M, w' \models \alpha
\end{array}
$$

Note that the only difference between **L** and **O** is that the (implicit) *if ... then* is replaced by *iff*. As a notational convenience, we often write $M \models \alpha$ if α is subjective, since the truth of α depends solely on the set of worlds M. Similarly, we write $w \models \alpha$ for objective α.

A set of sentences Γ *logically implies* a sentence α ($\Gamma \models \alpha$) iff for all worlds w and sets of worlds M, if $M, w \models \gamma$ for all $\gamma \in \Gamma$, then $M, w \models \alpha$. α is *valid* ($\models \alpha$) iff $\{\} \models \alpha$.

2.1 Some properties of *OL*

The reader is referred to [Lev90] for a sound and complete axiomatization of *OL*. Here we only touch on some of the main aspects of the logic.

It is well known that, given a globally accessible set of worlds M, the properties of **L** are precisely those of *weak S5* [HM85], that is, beliefs are closed under logical implication ($\models \mathbf{L}\alpha \wedge \mathbf{L}(\alpha \supset \beta) \supset \mathbf{L}\beta$) and positive as well as negative introspection ($\models \mathbf{L}\alpha \supset \mathbf{L}\mathbf{L}\alpha$ and $\models \neg\mathbf{L}\alpha \supset \mathbf{L}\neg\mathbf{L}\alpha$).

As far as the operator **O** is concerned, its properties are probably best explaine by noting its tight connection to the stable expansions of autoepistemic logic as proposed by Moore [Moo85].

Definition 2 (Moore) *Stable Expansions*
A set of sentences Γ is a **stable expansion** *of a set of sentences A iff Γ satisfies the fixed-point equation:*

$$
\Gamma = \{\alpha \mid \alpha \text{ is basic and } A \cup \{\mathbf{L}\beta \mid \beta \in \Gamma\} \cup \{\neg\mathbf{L}\beta \mid \beta \notin \Gamma\} \models_{taut} \alpha\},
$$

where $\models_{taut}$ denotes tautological consequence, that is, logical consequence of classical propositional logic.

Definition 3 *Belief Set*
A set of basic sentences is a **belief set** *for a set of worlds M iff $\Gamma = \{\alpha \mid M \models \mathbf{L}\alpha\}$.*

Although belief sets contain non-objective sentences, it turns out that it suffices to consider only the objective ones in order to distinguish between different belief sets.

Lemma 2.1 *Belief sets are uniquely determined by the objective sentences they contain.*

Proof: A proof can be found in [HM84]. ■

[3] Levesque considers the more general case of a first-order language with quantifying-in.

[4] We will freely use other connectives like $\wedge$, $\supset$ and $\equiv$, which should be understood as syntactic abbreviations of the usual kind.

[5] Possible-world semantics goes back to Kripke [Kri63]. Its application to epistemic notions is due to Hintikka [Hin62].

The following corollary will be useful in the next section.

Corollary 2.2 *Belief sets are uniquely determined by the objective clauses they contain.*

Proof : Follows directly from the fact that beliefs have equivalent conjunctive normal forms and that $\models \mathbf{L}(\bigwedge \alpha_i) \equiv \bigwedge \mathbf{L}\alpha_i$. ■

The next theorem demonstrates that *OL* subsumes Moore's autoepistemic logic provided we restrict ourselves to a finite set of premises (understood conjunctively).

Theorem 1 (Levesque) *For any basic α and any set of worlds M, $M \models \mathbf{O}\alpha$ iff the belief set of α is a stable expansion of $\{\alpha\}$.*[6]

3 The Logic OL^a

In this section we extend *OL* in a way that allows us to express the fact "this is all I know *about* a finite set of atomic propositions π". For that purpose, we add an infinite number of modal operators $\mathbf{O}\langle\pi\rangle$ for each finite set of atoms π to the language. If we refer to a set extensionally, we usually leave out the curly brackets. For example, we write $\mathbf{O}\langle p, q\rangle$ instead of $\mathbf{O}\langle\{p, q\}\rangle$.

Our task is now to define the semantics of $\mathbf{O}\langle\pi\rangle\alpha$, that is, given a set of worlds M, when is it the case that α is all M knows about the atoms in π? We approach this question by first defining a set of worlds $M|_\pi$ that has the same knowledge about π as M but knows nothing else. In a sense, $M|_\pi$ is obtained from M by forgetting everything that is irrelevant to π. With that construction, it seems that believing only α about a subject π at a set of worlds M reduces to believing only α (without any further qualifications) at $M|_\pi$. It turns out, however, that this definition sometimes has the effect that one believes only α about π without actually believing α, which seems counterintuitive.[7] This problem is caused, roughly, by the fact that an introspective agent who forgets not only loses knowledge but also gains knowledge, namely about his or her increased ignorance. In order to circumvent this problem, we simply add the restriction that α must be believed to the definition of $\mathbf{O}\langle\pi\rangle\alpha$. We now turn to the formal definitions.

$M|_\pi$ is defined as the set of all worlds that satisfy precisely the known objective sentences about π. The question is, of course, how to get at just those known sentences about π. First, by Corollary 2.2, it suffices to consider only the known *clauses* instead of arbitrary sentences. Furthermore, we only need to look at clauses that are minimal in the sense that they do not already follow from other known clauses:

Definition 4 *M-minimality*
Given a set of worlds M, a clause c is called **M-minimal** *iff $M \models \mathbf{L}c$ and for all clauses $c' \subsetneq c$, $M \not\models \mathbf{L}c'$.*

By restricting ourselves to M-minimal clauses, we rule out clauses that mention the subject matter but do not really tell us anything about it. For example, let the subject matter be p and assume all we know is q, that is, $M = \{w \mid w \models q\}$. Then we certainly also know $(p \vee q)$, which is not M-minimal because q is known as well. While $(p \vee q)$ mentions the subject matter p, it does so, in a sense, only accidentally, since it does not convey us what is really known about p, namely nothing. The only M-minimal clause mentioning p is $(p \vee \neg p)$, which gives us the right information.

In general, what is known about a subject matter π at a set of worlds M is captured by those M-minimal clauses that mention any of the atomic propositions in π. The only exception occurs when M is empty. In this case M knows every sentence and has, in particular, contradictory about the subject matter, which is characterized by $\Box$, the only M-minimal clause if M is empty. This leads us to the following definition.

Definition 5 *M-π-minimality*
Given a set of worlds M and a subject matter π, a clause c is called **M-π-minimal** *iff c is M-minimal and, in addition, $c = \Box$ or c contains either p or $\neg p$ for some $p \in \pi$.*

$M|_\pi$ is now simply the set of all worlds that satisfy all M-π-minimal sentences.

Definition 6
Given a set of worlds M and a subject matter π,

$$M|_\pi = \{w \mid w \models c \text{ for all } M\text{-}\pi\text{-minimal clauses } c\}.$$

Given $M|_\pi$, we are now able to characterize the meaning of $\mathbf{O}\langle\pi\rangle$ in terms $\mathbf{O}$ and $\mathbf{L}$:

$$M, w \models \mathbf{O}\langle\pi\rangle\alpha \iff M|_\pi, w \models \mathbf{O}\alpha \text{ and } M \models \mathbf{L}\alpha.$$

Logical implication and *validity* in OL^a are defined as in *OL*.

This completes the semantics of the extended logic OL^a.

3.1 Some properties of OL^a

So far, we have not obtained a complete axiomatization of OL^a. In the following, we present examples

[6] Levesque also shows that we can drop the condition that sentences are basic if we generalize the notion of stable expansions appropriately. Also, for reasons independent of this theorem, Levesque restricts himself to so-called maximal sets of worlds, a complication which we simply ignore here.

[7] An example is given in Section 3.1.

that demonstrate that the new concept $\mathbf{O}\langle\pi\rangle$ has indeed reasonable properties. (The proofs are deferred to the end of this subsection.) A more concise picture of what $\mathbf{O}\langle\pi\rangle$ is all about will emerge in the next section, where we show a strong connection between only-knowing-about and the ATMS.

Let α and β be sentences and p and q distinct atoms.

1. $\models\mathbf{O}\langle\pi\rangle\alpha \supset \mathbf{L}\alpha$,
   that is, if α is all I believe about π then I surely believe α. However, ...
2. $\not\models\mathbf{O}\langle\pi\rangle\alpha \supset \mathbf{O}\alpha$ (unless $\models\neg\mathbf{O}\langle\pi\rangle\alpha$),
   since I could also believe q for any atom not in π or α. On the other hand, if π contains all the atoms occurring in α, only-knowing and only-knowing-about coincide, that is, $\models\mathbf{O}\langle\pi\rangle\alpha \equiv \mathbf{O}\alpha$.
3. If $\models\alpha \equiv \beta$, then $\models\mathbf{O}\langle\pi\rangle\alpha \equiv \mathbf{O}\langle\pi\rangle\beta$.
   If all I believe about π is α, then the syntactic form of α is immaterial.
4. $\models\mathbf{O}\langle p\rangle(p \vee q) \supset (\neg\mathbf{L}p \wedge \neg\mathbf{L}q)$.
   This is because if I knew either p or q, then $p \vee q$ would be contingent information and thus, in either case, would not capture what I really know about p. Similarly,
5. $\models\mathbf{O}\langle p\rangle(p \vee q) \supset (\neg\mathbf{L}\neg p \wedge \neg\mathbf{L}\neg q)$.
   For if I knew $\neg p$, I would also know q because I know $(p \vee q)$. Thus $(p \vee q)$ would once again not capture what I really know about p. (Similarly, if I knew $\neg q$.)
6. $\models\mathbf{O}\langle p, q\rangle p \supset \mathbf{O}\langle q\rangle(q \vee \neg q)$.
   If all I know about p and q is p, then I know nothing about q.

   The following example treats a case where it is absurd to say this is all I know about p.
7. $\models\neg\mathbf{O}\langle p\rangle q$.
   In other words, something totally independent of p cannot be all I know about p. For example, it does not make sense to say that all I know about Tweety is that roses are red.
8. As Theorem 1 shows, only-knowing captures autoepistemic reasoning. The following example indicates that the weaker assumption of only-knowing-about in fact suffices for this purpose.

   Let p denote the proposition *Tweety flies*. Given the assumption that Tweety flies unless known otherwise, autoepistemic reasoning lets us conclude that Tweety flies. Formally,

   $$\models\mathbf{O}(\neg\mathbf{L}\neg p \supset p) \supset \mathbf{L}p.$$

   Later, if we discover that Tweety indeed does not fly, we retract our previous conclusion, that is,

   $$\models\mathbf{O}(\neg p \wedge (\neg\mathbf{L}\neg p \supset p)) \supset \neg\mathbf{L}p.$$

   However, it is intuitively clear that we need not require that the assumption is all we know to get the desired conclusion, but that the weaker requirement that this is all we know *about Tweety* suffices. And indeed, our formalization gives us just that. (The subject matter *about Tweety* in this case is simply $\{p\}$.)

   $\models\mathbf{O}\langle p\rangle(\neg\mathbf{L}\neg p \supset p) \supset \mathbf{L}p$.
   $\models\mathbf{O}\langle p\rangle(\neg p \wedge (\neg\mathbf{L}\neg p \supset p)) \supset \neg\mathbf{L}p$.
9. The following example concerns the case of sentences with multiple stable expansions. Again, the weaker requirement of only-knowing-about suffices to make the same distinctions as regular autoepistemic logic.

   $\models\mathbf{O}\langle p\rangle(\mathbf{L}p \supset p) \equiv [\mathbf{O}\langle p\rangle p \vee \mathbf{O}\langle p\rangle(p \vee \neg p)]$.
   In other words, believing only $\mathbf{L}p \supset p$ about p is the same as believing either only p or nothing about p.

Proofs:

1. $\models\mathbf{O}\langle\pi\rangle\alpha \supset \mathbf{L}\alpha$.
   Follows immediately from the definition of $\mathbf{O}\langle\pi\rangle$.
   It is worth noting that the property does not hold in general, if we omit the condition '$M\models\mathbf{L}\alpha$' from the definition of $\mathbf{O}\langle\pi\rangle$. Here is an example. Let $M^* = \{w \mid w\models q\}$. Then M^* knows absolutely nothing about p, that is, $M^*\models\mathbf{O}\langle p\rangle(p \vee \neg p)$ or, equivalently, $M^*\models\mathbf{O}\langle p\rangle\neg\mathbf{L}p$ because $M^*|_p = M_0$. Note that $M^*|_p\models\mathbf{O}\neg\mathbf{L}q$, since it knows nothing about q. However, $M^*\not\models\mathbf{L}\neg\mathbf{L}q$.
2. $\not\models\mathbf{O}\langle\pi\rangle\alpha \supset \mathbf{O}\alpha$ (unless $\models\neg\mathbf{O}\langle\pi\rangle\alpha$).
   Let $\alpha = p$, $\pi = \{p\}$, and let $M = \{w \mid w\models p \wedge q\}$. Since the only M-minimal clause that mentions p is p itself, $M|_\pi = \{w \mid w\models p\}$. Since $M\models\mathbf{L}p$ and $M|_\pi\models\mathbf{O}p$, we obtain $M\models\mathbf{O}\langle\pi\rangle p$, but $M\not\models\mathbf{O}p$. In fact, $M\models\mathbf{O}(p \wedge q)$.
3. If $\models\alpha \equiv \beta$, then $\models\mathbf{O}\langle\pi\rangle\alpha \equiv \mathbf{O}\langle\pi\rangle\beta$.
   Let $\models\alpha \equiv \beta$ and let M be a set of worlds such that $M\models\mathbf{O}\langle\pi\rangle\alpha$. We need to show that $M\models\mathbf{O}\langle\pi\rangle\beta$. (The reverse direction is completely symmetric and is omitted.)

   By assumption, $M\models\mathbf{L}\alpha$, that is, for all $w \in M$, $M, w\models\alpha$. Since $\models\alpha \equiv \beta$, we obtain that for all $w \in M$, $M, w\models\beta$, which implies that $M\models\mathbf{L}\beta$.

   Since, by assumption, $M|_\pi\models\mathbf{O}\alpha$, we obtain that for all worlds w, $w \in M|_\pi$ iff $M|_\pi, w\models\alpha$. Since $\models\alpha \equiv \beta$, this is the same as for all w, $w \in M|_\pi$ iff $M|_\pi, w\models\beta$ and, therefore, $M|_\pi\models\mathbf{O}\beta$.

   Since $M\models\mathbf{L}\beta$ and $M|_\pi\models\mathbf{O}\beta$, $M\models\mathbf{O}\langle\pi\rangle\beta$ follows.
4. $\models\mathbf{O}\langle p\rangle(p \vee q) \supset (\neg\mathbf{L}p \wedge \neg\mathbf{L}q)$.
   Let $M\models\mathbf{O}\langle p\rangle(p \vee q)$. Then $M|_p\models\mathbf{O}(p \vee q)$, that is, $M|_p = \{w \mid w\models(p \vee q)\}$. Assume that $M\models\mathbf{L}p$. Then $M|_p\models\mathbf{L}p$, since p is M-minimal. However, $M|_p$ contains a world w^* such that $w^*\models q$ yet $w^*\not\models p$, a contradiction.

Now assume that $M \models \mathbf{L}q$. Then none of the M-minimal clauses that mention p contains q as a literal. (In particular, $(p \vee q)$ is no longer M-minimal.) As noted before, $M|_p$ contains a world w^* such that $w^* \models q$ yet $w^* \not\models p$. By definition of $M|_p$, $w^* \models c$ for all M-minimal clauses c that mention p. Let w^{**} be exactly like w^* except that $w^{**} \models \neg q$. Note that $w^{**} \models c$ for all M-minimal clauses c that mention p, because none of those c contains the literal q. Thus $w^{**} \in M|_p$, but $w^{**} \not\models (p \vee q)$, a contradiction.

5. $\models \mathbf{O}\langle p\rangle(p \vee q) \supset (\neg\mathbf{L}\neg p \wedge \neg\mathbf{L}\neg q)$.
Let $M \models \mathbf{O}\langle p\rangle(p \vee q)$. Then $M|_p \models \mathbf{O}(p \vee q)$, that is, $M|_p = \{w \mid w \models (p \vee q)\}$. Assume $M \models \mathbf{L}\neg p$. Then, since $M \models \mathbf{L}(p \vee q)$, $M \models \mathbf{L}q$ follows, contradicting the previous property. Similarly, if we assume that $M \models \mathbf{L}\neg q$, then $M \models \mathbf{L}p$ follows, a contradiction.

6. $\models \mathbf{O}\langle p, q\rangle p \supset \mathbf{O}\langle q\rangle(q \vee \neg q)$.
Let $\pi = \{p, q\}$ and $M \models \mathbf{O}\langle\pi\rangle p$. Assume $M \not\models \mathbf{O}\langle q\rangle(q \vee \neg q)$. Thus $M|_q \not\models \mathbf{O}(q \vee \neg q)$ (since $M \models \mathbf{L}(q \vee \neg q)$) and, hence, $M|_q \neq M_0$ (the set of all worlds). In particular, there is a clause c^* which is M-minimal and mentions q and which is not a tautology. Note also that c^* does not contain the literal p because p itself is M-minimal. By the definition of $M|_\pi$, $M|_\pi \models \mathbf{L}c^*$. But $M|_\pi = \{w \mid w \models p\}$. Thus there is a world $w^* \in M|_\pi$ such that $w^* \not\models c^*$ (since c^* does not contain the literal p), a contradiction.

7. $\models \neg\mathbf{O}\langle p\rangle q$.
Assume, to the contrary, that there is an M such that $M \models \mathbf{O}\langle p\rangle q$. Then $M|_p \models \mathbf{O}q$, that is, $M|_p = \{w \mid w \models q\}$. Since $M \models \mathbf{L}q$, q is M-minimal and no M-minimal clause that mentions p contains q as a literal. Thus there exists a w^* that satisfies all M-minimal clauses mentioning p and $w^* \not\models q$. But $w^* \in M|_p$, a contradiction.

8. $\models \mathbf{O}\langle p\rangle(\neg\mathbf{L}\neg p \supset p) \supset \mathbf{L}p$.
Let $M \models \mathbf{O}\langle p\rangle(\neg\mathbf{L}\neg p \supset p)$. Then $M|_p \models \mathbf{O}(\neg\mathbf{L}\neg p \supset p)$ and thus $M|_p = \{w \mid w \models p\}$, that is, $M|_p \models \mathbf{L}p$. Since $M \subseteq M|_p$, $M \models \mathbf{L}p$ holds as well.

   $\models \mathbf{O}\langle p\rangle(\neg p \wedge (\neg\mathbf{L}\neg p \supset p)) \supset \neg\mathbf{L}p$.
   Similar to the previous case, if we assume that $M \models \mathbf{O}\langle p\rangle(\neg p \wedge (\neg\mathbf{L}\neg p \supset p))$, then $M|_p \models \mathbf{O}(\neg p \wedge (\neg\mathbf{L}\neg p \supset p))$ and thus $M|_p = \{w \mid w \models \neg p\}$, which implies that $M \models \mathbf{L}\neg p$ and $M \models \neg\mathbf{L}p$.

9. $\models \mathbf{O}\langle p\rangle(\mathbf{L}p \supset p) \equiv [\mathbf{O}\langle p\rangle p \vee \mathbf{O}\langle p\rangle(p \vee \neg p)]$.
First we prove that
$\models \mathbf{O}\langle p\rangle(\mathbf{L}p \supset p) \supset [\mathbf{O}\langle p\rangle p \vee \mathbf{O}\langle p\rangle(p \vee \neg p)]$.
Let $M \models \mathbf{O}\langle p\rangle(\mathbf{L}p \supset p)$. Then $M|_p \models \mathbf{O}(\mathbf{L}p \supset p)$. It is easy to see that there are only two cases: (a) $M|_p = \{w \mid w \models p\}$ and (b) $M|_p = M_0$. In case (a) we obtain $M \models \mathbf{L}p$, $M|_p \models \mathbf{O}p$, and, hence, $M \models \mathbf{O}\langle p\rangle p$. In case (b), $M \models \mathbf{L}(p \vee \neg p)$, $M|_p \models \mathbf{O}(p \vee \neg p)$, and, hence, $M \models \mathbf{O}\langle p\rangle(p \vee \neg p)$.
To prove
$\models [\mathbf{O}\langle p\rangle p \vee \mathbf{O}\langle p\rangle(p \vee \neg p)] \supset \mathbf{O}\langle p\rangle(\mathbf{L}p \supset p)$,
let us first assume that $M \models \mathbf{O}\langle p\rangle p$. Then $M \models \mathbf{L}p$ and $M|_p \models \mathbf{O}p$.
Therefore, $M \models \mathbf{L}(\mathbf{L}p \supset p)$ and $M|_p \models \mathbf{O}(\mathbf{L}p \supset p)$, which implies $M \models \mathbf{O}\langle p\rangle(\mathbf{L}p \supset p)$. Now let us assume that $M \models \mathbf{O}\langle p\rangle(p \vee \neg p)$. Then $M \models \mathbf{L}\neg\mathbf{L}p$ and, hence, $M \models \mathbf{L}(\mathbf{L}p \supset p)$. Also, since $M|_p \models \mathbf{O}(p \vee \neg p)$, $M|_p \models \mathbf{O}(\mathbf{L}p \supset p)$ holds as well. Again, $M \models \mathbf{O}\langle p\rangle(\mathbf{L}p \supset p)$ follows. ■

3.2 Levesque's Proposal

As mentioned before, Levesque suggests a definition of $\mathbf{O}\langle\pi\rangle$ in [Lev90], yet without analyzing it in any depth. We now briefly discuss Levesque's proposal and point out some of its problems.

In our notation and restricted to the propositional case, Levesque's definition goes as follows:

$$M \models \mathbf{O}\langle\pi\rangle\alpha \iff \text{for all } w,\ w \in M \text{ iff } M, w \models \alpha \text{ and there is a } w' \in M \text{ such that for all atoms } p, \text{ if } w(p) \neq w'(p), \text{ then } p \in \pi.$$

The extra condition (compared to the definition of $\mathbf{O}$) amounts to saying that w must be an element of M unless it violates knowledge about something other than π.

At first glance, this definition looks quite appealing because of its simplicity. It also shares many of the properties with our definition of $\mathbf{O}\langle\pi\rangle$. Unfortunately, it also has some rather counterintuitive properties. Here are two examples. In the following, we abbreviate the condition "*... and there is a $w' \in M$ such that for all atoms p, if $w(p) \neq w'(p)$, then $p \in \pi$*" by $(*)$.

1. Let $M = \{w \mid w \models q\}$. Intuitively, M knows nothing about p. However, according to Levesque's definition, $M \models \mathbf{O}\langle p\rangle q$, that is, M claims to know something totally irrelevant about p.
   Proof: First, let $w \in M$. Obviously $w \models (p \vee q)$. The condition $(*)$ is satisfied simply by choosing $w' = w$. If $w \notin M$, then $w \not\models q$ and we are done.

2. Let M be as before. Then $M \models \mathbf{O}\langle p\rangle(p \supset q)$. This seems even more misleading, since M's alleged knowledge about p seems at least plausible.
   Proof: For any $w \in M$, $w \models q$ and w satisfies $(*)$. Conversely, let $w \notin M$. We have to show that if $w \models (p \supset q)$, then $(*)$ is violated. Thus let us assume that $w \models (p \supset q)$. Then $w \not\models p$ and $w \not\models q$. Therefore, any $w' \in M$ is such that $w(q) \neq w'(q)$, but $q \notin \{p\}$.

While it seems possible to filter out occurrences of ex-

ample (1) by adding a suitable restriction that forces α to be relevant to π, case (2) cannot be resolved by looking at α and π alone.

4 Computing all that is known about some subject matter

Having defined what it means for α to be all an agent knows about a certain subject matter π, it seems natural to ask how to compute α from the agent's knowledge base (KB). In this section we will provide an answer for the special yet important case of *objective* KBs, that is, no modal operators are allowed in the KB. This case is particularly interesting because computing all that is known about π reduces to abductive reasoning and, in particular, to computing *explanations* in the sense of an ATMS [deK86] for the atoms in π and their negations. Thus we are able to relate only-knowing-about to more familiar notions in knowledge representation.

Following [Lev90], a sentence α is said to explain a sentence β just in case $\alpha \supset \beta$ is believed and $\neg\alpha$ is not believed. If $\mathbf{L}$ is our model of belief and the beliefs are represented by an objective KB,[8] Levesques has shown that his definition coincides with previous definitions of abduction such as Poole's [Poo88], where α explains β iff $\mathrm{KB} \cup \{\alpha\} \models \beta$ and $\mathrm{KB} \cup \{\alpha\}$ is consistent.

An ATMS then computes the set of all simplest explanations[9] for a given KB and β.

As an example, let us consider a simple minded medical KB which only knows that hepatitis causes jaundice and that the patient has a fever, represented as

$$\mathrm{KB} = \{(\mathit{hepatitis} \supset \mathit{jaundice}), \mathit{fever}\}.$$

An ATMS, asked for the explanations of *jaundice*, returns both *hepatitis* and the trivial explanation *jaundice* itself, denoted as

$$\mathbf{atms}[\mathrm{KB}, \mathit{jaundice}] = \{\mathit{jaundice}, \mathit{hepatitis}\}.$$

Notice that $\mathit{hepatitis} \supset \mathit{jaundice}$ is all that is known about jaundice according to our definition. We can reconstruct this information systematically using the output from the ATMS as

$$\alpha = \bigwedge_{c \in \mathbf{atms}[\mathrm{KB}, \mathit{jaundice}]} (c \supset \mathit{jaundice})$$

It is easy to verify that

$$\models \mathbf{O}\mathrm{KB} \supset \mathbf{O}(\mathit{jaundice})\alpha$$

because
$\alpha \equiv (\mathit{jaundice} \supset \mathit{jaundice}) \wedge (\mathit{hepatitis} \supset \mathit{jaundice})$
$\equiv (\mathit{hepatitis} \supset \mathit{jaundice})$.
In the following, we examine the general case of computing all that is known about a subject matter using an ATMS (Theorem 2).

To simplify matters, we assume that the subject matter π consists of a single atom p. All of the following results generalize in a straightforward way to arbitrary sets due to the following lemma:

Lemma 4.1 *Let π_1 and π_2 be two sets of atoms and α and β objective sentences. Then*
$\models \mathbf{O}\langle\pi_1\rangle\alpha \wedge \mathbf{O}\langle\pi_2\rangle\beta \supset \mathbf{O}\langle\pi_1 \cup \pi_2\rangle(\alpha \wedge \beta)$.

Proof : Let M be a set of worlds such that $M \models \mathbf{O}\langle\pi_1\rangle\alpha \wedge \mathbf{O}\langle\pi_2\rangle\beta$. By the semantics of $\mathbf{O}\langle\pi_1\rangle$ and $\mathbf{O}\langle\pi_2\rangle$, $M \models \mathbf{L}\alpha$ and $M \models \mathbf{L}\beta$ and hence $M \models \mathbf{L}(\alpha \wedge \beta)$. Let $\pi = \pi_1 \cup \pi_2$. We still need to prove that $M|_\pi \models \mathbf{O}(\alpha \wedge \beta)$.

First we show that $M|_\pi = M|_{\pi_1} \cap M|_{\pi_2}$. If $M = \{\}$ then $M|_\pi = M|_{\pi_1} = M|_{\pi_2} = \{\}$ and we are done. Now assume that M is not empty. Then $M|_\pi$, $M|_{\pi_1}$, and $M|_{\pi_2}$ are not empty either. To show that $M|_\pi \models \mathbf{O}(\alpha \wedge \beta)$, let w be any world. $w \in M|_\pi$ iff $w \models c$ for all M-minimal clauses c that mention some $p \in \pi$ iff $w \models c$ for all M-minimal clauses c that mention some $p \in \pi_1$ and $w \models c$ for all M-minimal clauses c that mention some $p \in \pi_2$ iff $w \in M|_{\pi_1} \cap M|_{\pi_2}$.

By assumption, $M|_{\pi_1} \models \mathbf{O}\alpha$ and $M|_{\pi_2} \models \mathbf{O}\beta$. Since α and β are objective, $M|_{\pi_1} = \{w \mid w \models \alpha\}$ and $M|_{\pi_2} = \{w \mid w \models \beta\}$. Therefore $M|_\pi = \{w \mid w \models \alpha\} \cap \{w \mid w \models \beta\}$. From this it is straightforward to show that $M|_\pi \models \mathbf{O}(\alpha \wedge \beta)$. ■

Notation: For any clause c, let $\bar{c}$ denote the complement of c, that is, the conjunction of the complements of the literals contained in c. Also, from now on KB denotes a finite set of clauses. Whenever the formalism requires a sentence instead of a set, we write KB as well and mean the conjunction of all the clauses in the knowledge base. Let $\Re[\mathrm{KB}] = \{w \mid w \models \mathrm{KB}\}$, that is $\Re[\mathrm{KB}]$ represents the beliefs that follow from KB.

The following definitions are mainly adapted from [Lev89].

First we need to define what it means for a sentence to be a simplest (or minimal) explanation.

Definition 7 *Simplicity*
The set of literals contained in an objective sentence is defined as follows:
$\mathrm{LITS}(\Box) = \{\}$; $\mathrm{LITS}(p) = \{p\}$;
$\mathrm{LITS}(\neg\alpha) = \{\bar{l} \mid l \in \mathrm{LITS}(\alpha)\}$;
$\mathrm{LITS}(\alpha \wedge \beta) = \mathrm{LITS}(\alpha) \cup \mathrm{LITS}(\beta)$.
α is simpler than β ($\alpha \prec \beta$) iff $\mathrm{LITS}(\alpha) \subsetneq \mathrm{LITS}(\beta)$.

Definition 8 *Explanations and minimal explanations*
Let α and β be objective sentences. Then

[8] The beliefs represented by KB are characterized by the set of all worlds that satisfy KB.

[9] Roughly, an explanation α is simpler than β if the literals in α are contained in β.

1. α $\text{expl}_{\mathbf{L}}$ β *wrt.* KB *iff* $\Re[\text{KB}] \models \mathbf{L}(\alpha \supset \beta) \wedge \neg\mathbf{L}\neg\alpha$.

2. α $\text{min\_expl}_{\mathbf{L}}$ β *wrt.* KB *iff* α $\text{expl}_{\mathbf{L}}$ β *wrt.* KB *and for no* $\alpha^* \prec \alpha$, α^* $\text{expl}_{\mathbf{L}}$ β *wrt.* KB.

We now turn to Reiter and deKleer's formalization of an ATMS [RdK87].

Definition 9 *An ATMS*

1. $\text{IMPS}[\text{KB}] = \{c \mid c$ *is a clause and* $\text{KB} \models c\}$.

2. $\mu\text{IMPS}[\text{KB}] = \{c \mid c \in \text{IMPS}[\text{KB}]$ *and for no* $c' \subsetneq c$, $c' \in \text{IMPS}[\text{KB}]\}$.[10]

3. *Let* c *be a clause. Then* $\mathbf{atms}[\text{KB}, c] = \{(q_1 \wedge \ldots \wedge q_k) \mid k \geq 0$ and $(\overline{q_1} \vee \ldots \vee \overline{q_k} \vee c) \in \mu\text{IMPS}[\text{KB}]\}$.[11]

Note that **atms** returns explanations in a special syntactic form, namely as conjunctions of literals. Levesque's definition of simplest explanations, on the other hand, does not worry about how an explanation is represented. Nevertheless, as shown by Levesque, the two notions are equivalent at the level of propositions,[12] that is, the propositions expressed by $\mathbf{atms}[\text{KB}, c]$ are the same as the propositions expressed by the simplest explanations of c with respect to KB. Given Definition 9, we are now able to prove that all that is known about p by KB is completely characterized by $\mathbf{atms}[\text{KB}, p]$ together with $\mathbf{atms}[\text{KB}, \neg p]$. First, we note a straightforward connection between IMPS[KB] and the notion of $\Re[\text{KB}]$-minimality (Definition 4).

Lemma 4.2 *For any clause* c, $c \in \mu\text{IMPS}[\text{KB}]$ *iff* c *is* $\Re[\text{KB}]$*-minimal.*

Proof: Note that $\Re[\text{KB}] \models \mathbf{L}\alpha$ iff $\text{KB} \models \alpha$ for all objective sentences α. The lemma follows now immediately from the definitions of $\Re[\text{KB}]$-minimality and $\mu\text{IMPS}[\text{KB}]$. ■

Theorem 2 *Let* p *be an atom,* $\Gamma_p = \mathbf{atms}[\text{KB}, p]$, *and* $\Gamma_{\overline{p}} = \mathbf{atms}[\text{KB}, \neg p]$. *Then*

$$\models \text{OKB} \supset \mathbf{O}\langle p\rangle[\bigwedge_{c \in \Gamma_p} (c \supset p) \wedge \bigwedge_{c \in \Gamma_{\overline{p}}} (c \supset \neg p)].$$

Proof:

Let $\alpha = \bigwedge_{c \in \Gamma_p}(c \supset p) \wedge \bigwedge_{c \in \Gamma_{\overline{p}}}(c \supset \neg p)$. (Note that $(c \supset p)$ and $(c \supset \neg p)$ really are clauses, since c is a conjunction of literals.) First, it is not hard to show that both Γ_p and $\Gamma_{\overline{p}}$ are finite so that α is in fact a well-defined objective sentence. Let $M = \{w \mid w \models \alpha\}$. Then $M = \{w \mid w \models (c \supset p)$ for all $c \in \Gamma_p\} \cap \{w \mid w \models (c \supset \neg p)$ for all $c \in \Gamma_{\overline{p}}\}$. By Lemma 4.2, $\Gamma_p = \{c \mid (c \supset p)$ *is* $\Re[\text{KB}]$-minimal$\}$ and $\Gamma_{\overline{p}} = \{c \mid (c \supset \neg p)$ *is* $\Re[\text{KB}]$-minimal$\}$. Note that, by Definition 6,

$$\Re[\text{KB}]|_p = \{w \mid w \models (c \supset p) \text{ for all } \Re[\text{KB}]\text{-minimal } (c \supset p)\} \cap \{w \mid w \models (c \supset \neg p) \text{ for all } \Re[\text{KB}]\text{-minimal } (c \supset \neg p)\}$$

Thus $\Re[\text{KB}]|_p = M$ and, hence, $\Re[\text{KB}]|_p \models \mathbf{O}\alpha$. Obviously, $M \models \mathbf{L}\alpha$ holds as well and, therefore, $M \models \mathbf{O}\langle p\rangle\alpha$.

[Note that in the case where KB is consistent and $\text{KB} \models p$, $\mathbf{atms}[\text{KB}, p] = \{\neg\Box\}$, in which case $\alpha \equiv (\neg\Box \supset p) \equiv p$ (similarly for $\text{KB} \models \neg p$). If KB is inconsistent, then $\mathbf{atms}[\text{KB}, p] = \{\}$. Thus α is the empty clause, represented as $\Box$.] ■

5 The Logic OB^a

While the previous section demonstrates how only-knowing-about can in principle be computed using an ATMS, the result also entails that the computational cost is considerable since the ATMS is inherently intractable [SL90].

The source of this complexity lies in the very powerful model of belief that we adopted, which requires belief to be closed under modus ponens, that is, $\models \mathbf{L}\alpha \wedge \mathbf{L}(\alpha \supset \beta) \supset \mathbf{L}\beta$. In particular, this means that even computing the objective beliefs of $\Re[\text{KB}]$ is intractable. Levesque has called this type of belief *implicit belief* and contrasts it with *explicit belief*, which should take into account a reasoner's limited resources [Lev84]. In [Lev89], Levesque suggests a particular form of explicit belief,[13] which does not only yield a tractable deductive reasoner, but which lends itself to the definition of a tractable form of abduction or limited ATMS. Given this result, we are able to show that an agent can efficiently compute all that he or she explicitly knows about some subject matter.

First we modify the logic OL^a in order to incorporate the notion of explicit belief along the lines of [Lev89].

For this purpose, we replace the operator **L** with a new operator **B** (for explicit belief). **B** and **O** in this new logic are variants of the same operators in [LL88]. Also **B** extends Levesque's notion of explicit belief in [Lev89] (which, in turn, is a variant of Levesque's logic of explicit belief in [Lev84]) by allowing arbitrary nestings of modal operators.

The semantics of this new logic OB^a is defined just like the semantics of OL^a except that **B** and **O** are inter-

[10] In the terminology of Reiter and deKleer, $\mu\text{IMPS}[\text{KB}]$ contains the *prime implicants* of KB.

[11] Actually, given deKleer's original formulation of an ATMS [deK86], this came out as a theorem in [RdK87].

[12] The proposition expressed by a sentence α can be thought of as the set of worlds that satisfy α.

[13] This is a variant of the type of explicit belief defined in [Lev84].

preted with respect to so-called *situations* instead of worlds and $\mathbf{O}\langle\pi\rangle$ is defined in terms of **B** and **O**. While worlds assign a truth value to every atomic proposition, situations, in contrast, assign truth values to all the *literals* with the restriction that either p or $\neg p$ is assigned **true**. This gives us the following important difference between worlds and situations: while a literal and its complement always have complementary truth values at a world, they may both have the value **t** at a situation.

Definition 10 *Situations*
A situation s is a function s : Literals $\longrightarrow \{\mathbf{t}, \mathbf{f}\}$ such that $s(\Box) = \mathbf{f}$ and for every atom p, either $s(p) = \mathbf{t}$ or $s(\neg p) = \mathbf{t}$.

The definitions for M-minimality, M-π-minimality, and $M|_\pi$ carry over from implicit belief to explicit belief in the obvious way. Note that those definitions rely on the fact that belief sets are uniquely determined by the objective clauses they contain. Lemma 5.1 below verifies that this is indeed the case for explicit belief as well.

Definition 11 *M-minimality*
Given a set of situations M, a clause c is called M-**minimal** *iff $M \models \mathbf{B}c$ and for all clauses $c' \subsetneq c$, $M \not\models \mathbf{B}c'$.*

Definition 12 *M-π-minimality*
Given a set of situations M and a subject matter π, a clause c is called M-π-**minimal** *iff c is M-minimal and, in addition, $c = \Box$ or c contains either p or $\neg p$ for some $p \in \pi$.*

Definition 13
Given a set of situations M and a subject matter π,
$M|_\pi = \{s \mid s$ *is a situation and* $s \models c$ *for all M-π-minimal clauses c}.*

Since situations assign independent truth values to literals and their complements, the semantic rules for arbitrary sentences must define truth support for sentences and their negations. Let p be an atom, α and β arbitrary sentences, s a situation, and M a set of situations. $M|_\pi$ is defined as in Section 3 except that worlds are replaced by situations.

$$
\begin{array}{lcl}
M, s \models p & \Longleftrightarrow & s(p) = \mathbf{t} \\
M, s \models \neg p & \Longleftrightarrow & s(\neg p) = \mathbf{t} \\
M, s \models \neg\neg\alpha & \Longleftrightarrow & M, s \models \alpha \\
M, s \models \alpha \vee \beta & \Longleftrightarrow & M, s \models \alpha \text{ or } M, s \models \beta \\
M, s \models \neg(\alpha \vee \beta) & \Longleftrightarrow & M, s \models \neg\alpha \text{ and } M, s \models \neg\beta \\
M, s \models \mathbf{B}\alpha & \Longleftrightarrow & \text{for all } s' \in M,\ M, s' \models \alpha \\
M, s \models \mathbf{O}\alpha & \Longleftrightarrow & \text{for all } s',\ s' \in M \text{ iff } M, s' \models \alpha \\
M, s \models \mathbf{O}\langle\pi\rangle\alpha & \Longleftrightarrow & M|_\pi, s \models \mathbf{O}\alpha \text{ and } M, s \models \mathbf{B}\alpha
\end{array}
$$

Let Φ be any of the operators **B**,**O**, or $\mathbf{O}\langle\pi\rangle$.

$$M, s \models \neg\Phi\alpha \quad \Longleftrightarrow \quad M, s \not\models \Phi\alpha$$

Since we are interested in situations only as far as explicit belief is concerned, we define truth and logical implication, as usual, with respect to worlds only. Let w be a world and M a set of situations. A sentence α is said to be true at w and M just in case $M, w \models \alpha$. For a set of sentences Γ, $\Gamma \models \alpha$ iff for all worlds w and sets of situations M, if $M, w \models \gamma$ for all $\gamma \in \Gamma$, then $M, w \models \alpha$. Validity ($\models \alpha$) is defined as usual as $\{\} \models \alpha$.

5.1 Some properties of OB^a

As said before, explicit belief (**B**) and explicitly only-believing (**O**) are variants of similar notions in [LL88] (and also extensions of [Lev84, Lev89]). Thus we will not discuss the properties of **B** and **O** in detail here. What makes implicit belief interesting for our purposes is that it allows for tractable reasoning in the sense that, if we have an objective KB and an objective α, both in conjunctive normal form, then deciding whether $\mathbf{B}\alpha$ is logically implied by **B**KB is tractable [Lev89]. The algorithm is very simple and relies on the following fact. $\mathbf{B}\alpha$ is logically implied by **B**KB iff for every clause c in α one of the following properties holds:

1. c is tautologous, that is, it contains complementary literals.
2. There is a clause c' in KB such that $c' \subseteq c$.

The corresponding algorithm, of course, can be computed in time $O(|\text{KB}|, |\alpha|)$.[14] Notice that modus ponens is ruled out as a valid inference rule, that is, an agent who believes p and $p \supset q$ does not necessarily believe q.

Let us now turn to $\mathbf{O}\langle\pi\rangle$ in the context of explicit belief. Our approach to defining the meaning of $\mathbf{O}\langle\pi\rangle$ presumes that the beliefs at a set of situations are uniquely determined by the objective clauses they contain, just as in the case of implicit belief. The following lemma verifies that this is indeed the case.

Lemma 5.1 *The beliefs at a set of situations M are uniquely determined by the clauses that are believed at M.*

Proof : It suffices to prove that (1) the beliefs at M are uniquely determined by the objective sentences that are believed at M, (2) every belief can be transformed into an equivalent conjunctive normal form, and (3) $\models \mathbf{B} \bigwedge \alpha_i \equiv \bigwedge \mathbf{B}\alpha_i$. Proofs of these properties in the case of the logic of [LL88] carry over to OB^a in a straightforward way. ■

In Section 3.1, we proved properties of $\mathbf{O}\langle\pi\rangle$ in the case of implicit belief. Except for property 3, all of those hold in the case of explicit belief as well. In fact,

[14] Apart from testing for tautologous clauses, the algorithm in fact computes *tautological entailment*, a form of relevance logic [AB75, Dun76].

the proofs carry over simply by replacing *worlds* by situations and **L** by **B**.

To see that property 3 no longer holds, let $\alpha = p \wedge (p \supset q)$ and $\beta = p \wedge q$. Obviously, $\models \alpha \equiv \beta$. Let $M = \{s \mid s \text{ is a situation and } s \models \alpha\}$. It is easy to see that $M \models \mathbf{O}\langle p, q\rangle \alpha$, but M contains a situations s^* such that $s \models p$, $s \models \neg p$, and $s \not\models q$. Thus $M \not\models \mathbf{B}\beta$ and, hence, $M \not\models \mathbf{O}\langle p, q\rangle \beta$.

The following weakened form of property 3 holds, however, in the case of explicit belief:

3' If for any situation s and any set of situations M, $M, s \models \alpha$ iff $M, s \models \beta$, then $\models \mathbf{O}\langle\pi\rangle\alpha \equiv \mathbf{O}\langle\pi\rangle\beta$. (The proof is very similar to the proof of 3.)

5.2 Computing all that is known explicitly about some subject matter

In order to compute what an objective KB knows about some subject matter based on explicit belief, we use Levesque's definition of a limited form of abductive explanation, which is obtained from Definition 8 simply be replacing **L** with **B**. Also, we need to characterize the beliefs of an objective KB now in terms of situations rather than worlds. Therefore, let

$$\Re[\mathrm{KB}] = \{s \mid s \text{ is a } \textbf{situation} \text{ and } s \models \mathrm{KB}\}.$$

Definition 14 (Levesque) *Explanations and minimal explanations for explicit belief*
Let α and β be objective sentences. Then

1. α $\mathrm{expl}_{\mathbf{B}}$ β *wrt.* KB *iff* $\Re[\mathrm{KB}] \models \mathbf{B}(\alpha \supset \beta) \wedge \neg\mathbf{B}\neg\alpha$.
2. α $\mathrm{min\_expl}_{\mathbf{B}}$ β *wrt.* KB *iff* α $\mathrm{expl}_{\mathbf{B}}$ β *wrt.* KB *and for no* $\alpha^* \prec \alpha$, α^* $\mathrm{expl}_{\mathbf{B}}$ β *wrt.* KB.

Similarly, Levesque defines a limited version of an ATMS by replacing IMPS with EXPS, which is the set of clauses explicitly believed at $\Re[\mathrm{KB}]$.

Definition 15 (Levesque) *A limited ATMS*

1. $\mathrm{EXPS}[\mathrm{KB}] = \{c \mid c \text{ is a tautologous clause or } \exists c^* \in \mathrm{KB}, c^* \subseteq c\}$.
2. $\mu\mathrm{EXPS}[\mathrm{KB}] = \{c \mid c \in \mathrm{EXPS}[\mathrm{KB}] \text{ and for no } c' \subsetneq c,\ c' \in \mathrm{EXPS}[\mathrm{KB}]\}$.
3. *Let c be a clause. Then* $\mathtt{lim\_atms}[\mathrm{KB}, c] = \{(q_1 \wedge \ldots \wedge q_k) \mid k \geq 0 \text{ and } (\overline{q_1} \vee \ldots \vee \overline{q_k} \vee c) \in \mu\mathrm{EXPS}[\mathrm{KB}]\}$.[15]

As in the case of implicit belief, Levesque shows that minimal explanations and the result of `lim_atms` coincide if viewed as propositions [Lev89]. `lim_atms`, while considerably weaker than `atms`, has the benefit of being efficiently computable.

Theorem 3 (Levesque)
For any clause c, $\mathtt{lim\_atms}[\mathrm{KB}, c]$ *is computable in* $O(|\mathrm{KB}| \times |c|)$.

Coming back to our original goal of defining a tractable notion of only-knowing-about, we note that `lim_atms` gives us the right characterization of what is known about some atom p (we will look at an arbitrary π in a moment) for a given KB, just as `atms` does in the case of implicit belief.

Theorem 4 *Let p be an atom,* $\Gamma_p = \mathtt{lim\_atms}[\mathrm{KB}, p]$, *and* $\Gamma_{\overline{p}} = \mathtt{lim\_atms}[\mathrm{KB}, \neg p]$. *Then*

$$\models \mathbf{O}\mathrm{KB} \supset \mathbf{O}\langle p\rangle[\bigwedge_{c \in \Gamma_p} (c \supset p) \wedge \bigwedge_{c \in \Gamma_{\overline{p}}} (c \supset \neg p)].$$

Proof : The proof is a straightforward adaptation of the proof of the analogous result for implicit belief (Theorem 2). ■

More importantly, given the previous two theorems, it is not hard to show that all that is known about p is computable in time linear in the size of KB.

Theorem 5 *All that is known about p with respect to* KB *is computable in* $O(|\mathrm{KB}|)$.

Proof : By Theorem 3, Γ_p and $\Gamma_{\overline{p}}$ can be computed in time $O(|\mathrm{KB}|)$. Also, the size of

$$\alpha = \bigwedge_{c \in \Gamma_p} (c \supset p) \wedge \bigwedge_{c \in \Gamma_{\overline{p}}} (c \supset \neg p)$$

is bounded by the size of KB, that is, α can be constructed in time $O(|\mathrm{KB}|)$. ■

The general case of an arbitrary subject π is not much harder. First, note that Lemma 4.1 also holds in the case of OB^a. Thus, in order to compute what is known about π we only need to compute what is known about each $p \in \pi$ and conjoin the results. This can be done in $O(|\mathrm{KB}| \times |\pi|)$.

Finally, it is perhaps instructive to look at an example that clearly exhibits the reasons why it is so easy to compute what is known about π.

Example 5.1 Let $\mathrm{KB} = \{p, (s \supset q), (p \wedge q) \supset r\}$ and let $\pi = \{r\}$. Then $\mathbf{O}\mathrm{KB} \supset \mathbf{O}\langle\pi\rangle[(p \wedge q) \supset r]$, that is, in order to find out what is known about r we only need to go through the clauses (which are written as implications here) and collect those that mention r. Note that under implicit belief, we would also have to include $(s \supset r)$ since $\mathbf{O}\mathrm{KB} \supset \mathbf{L}(s \supset r)$. In other words, we would have to apply full resolution to find all the relevant clauses.

[15] Levesque considers the more general case of objective sentences in conjunctive normal form as the second argument to `lim_atms`, which he calls abd.

6 Summary and future work

In this paper, we proposed a formalization of only-knowing-about, that is, the notion that something is all an agent knows about some subject matter, where *subject matter* was taken to be a set of atoms in a propositional logic. We explored some of the logical properties of this notion and were able to show a tight connection to deKleer's ATMS. Furthermore, by weakening the underlying model of belief, we arrived at a form of only-knowing-about that is efficiently computable.

As for future work, OL^a needs an axiomatization to really understand all of the properties of only-knowing-about. Furthermore, the logic needs to be generalized to the first-order case. This is not at all straightforward, since the propositional case relies heavily on the fact that we can restrict ourselves to clauses rather than arbitrary sentences, an assumption that does not work in the first-order case. Another issue is the multi-agent scenario, where only-knowing-about plays an important role as indicated in the introduction. However, so far there is not even a multi-agent formalization of only-knowing!

Acknowledgements

I would like to thank Wiktor Marek, who suggested the title of the paper as the title of a talk on preliminary results of this work.

References

[AB75] Anderson, A. R. und Belnap, N. D., *Entailment, the logic of relevance and necessity.* Princeton University Press, 1975.

[deK86] de Kleer, J., An assumption-based TMS. *Artificial Intelligence*, **28**(2), 1986, pp. 127–162.

[Dun76] Dunn, J. M., Intuitive Semantics for First-Degree Entailments and Coupled Trees. *Philosophical Studies* **29**, 1976, pp. 149–168.

[HM84] Halpern, J. Y. and Moses, Y. O., Towards a Theory of Knowledge and Ignorance: Preliminary Report, in Proceedings of The Non-Monotonic Workshop, New Paltz, NY, 1984, pp.125–143.

[HM85] Halpern, J. Y. and Moses, Y. O., A Guide to the Modal Logics of Knowledge and Belief. *Proc. of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, 1985, pp. 480–490.

[Hin62] Hintikka, J., *Knowledge and Belief: An Introduction to the Logic of the Two Notions.* Cornell University Press, 1962.

[Kri63] Kripke, S. A., Semantical considerations on modal logic. *Acta Philosophica Fennica* **16**, pp. 83–94, 1963.

[LL88] Lakemeyer, G. and Levesque, H. J., A Tractable Knowledge Representation Service with Full Introspection. *Proc. of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, Asilomar, California, 1988, pp. 145–159.

[Lak92] All I know about Tweety, to appear in: *Proc. of the 2nd International Workshop on Nonmonotonic and Inductive Logics*, Lecture Notes in Computer Science, Springer Verlag, 1992.

[Lev84] Levesque, H. J., A Logic of Implicit and Explicit Belief. *Proc. of the 4th Natinal Conference on Artificial Intelligence (AAAI-84)*, Austin, TX, 1984, pp. 198–202.

[Lev89] Levesque, H. J., A knowledge-level account of abduction. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pp. 1061–1067, Detroit, MA, 1989.

[Lev90] Levesque, H. J., All I Know: A Study in Autoepistemic Logic. *Artificial Intelligence*, North Holland, **42**, 1990, pp. 263–309.

[Moo85] Moore, R., Semantical Considerations on Nonmonotonic Logic. *Artificial Intelligence* **25**, 1985, pp. 75–94.

[Poo88] Poole, D., A methodology for using a default and abductive reasoning system. Thechnical Report. Department of Computer Science, University of Waterloo, Waterloo, Ontario, 1988.

[RdK87] Reiter, R. und de Kleer, J., Foundations of assumption-based truth maintenance systems: preliminary report. *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, Seattle, WA, 1987, pp. 183–188.

[SL90] Selman, B. and Leveque H. J., Abductive and default reasoning: a computational core. *Proceedings of the Eigth National Conference on Artificial Intelligence (AAAI-90)*, Boston, MA, 1990, pp. 343–348.

Representing Defaults as Sentences with Reduced Priority

Mark Ryan
Department of Computing
Imperial College
London SW7 2BZ, UK
E-mail: mdr@doc.ic.ac.uk

Abstract

We distinguish between two ways of thinking about defaults. The first way, in which defaults augment known premises by 'strengthening' the underlying logic, is the traditional approach taken by most existing formalisms. In the second way, defaults are represented in the set of premises, but obtain their default status by having a reduced priority relative to the known premises. In this paper we:

1. Compare and contrast the approaches. We argue that the second approach makes for simpler representation of defaults and their interactions.

2. Describe a syntax and semantics for the second, less well-known approach; we introduce the notion of *ordered theory presentation* (OTP) to represent theories with defaults.

3. Show how ordered theory presentations can represent familiar examples of interacting defaults in an intuitively clear and simple way; we give the Tweety example and the Yale Shooting example. We also show that the OTP framework is particularly well suited to inheritance examples.

4. Show formal properties of OTPs, in particular cumulativity, and suggest connections with circumscription.

5. Show how OTPs may be used to model *belief revision* and compare the result with the standard theory.

1 Introduction

Most systems for reasoning with defaults treat them as a way of *strengthening the underlying logic*. For example, in circumscription defaults are represented by the policy of minimising certain predicates. Models of the circumscribed theory are those models of the original theory which have minimal extensions of those predicates. In particular, anything that can be proved without the defaults (i.e., without the minimisation) can also be proved with them; thus, the process of minimisation strengthens the deductive power of the logic. The same is true in, for example, negation as failure viewed as a default system; the default mechanism (in which the defaults are the negations of atomic formulas) allows us to derive more from a set of clauses than is classically derivable.

There is another view of defaults which is less widely known, although it has been described before [Bib85, Bre89, Poo88]. Whereas on the first view we had too few consequences of a theory, and used the default mechanism to add to them, on the second view we have too many consequences and the default mechanism reduces their number. In the second view, defaults are represented as *sentences in the theory* instead of as a means of augmenting the logic. The set of facts together with the set of defaults is in general contradictory. But the defaults are assigned a lower status, or reduced priority, than the other more certain sentences in the theory; this avoids contradictory conclusions. Much of this paper will flesh out both the syntax and the semantics of this 'reduced priority'.

We consider that the second view of defaults is preferable. Firstly, it provides a clearer way of specifying the default information. The fact that defaults are expressed as ordinary sentences using the full range of logical operators obviates the need for coding tricks which are often necessary in, for example, circumscription. Secondly, it treats defaults as part of the knowledge being represented, instead of as part of the logic. This gives improved knowledge representation.

In this paper we describe a system for representing defaults which falls into the second view of the two described above. In that system, a theory is presented as a partially-ordered set of sentences. (The exact definition is given in §2.) All of the sentences which we wish to represent are included in this set. That some of them are defaults and some are not is represented by their position in the ordering. The lower a sentence is in the ordering, the less of a default it is, and the minimal sentences are those which are not defaults at all. In the system we describe, we can have several levels

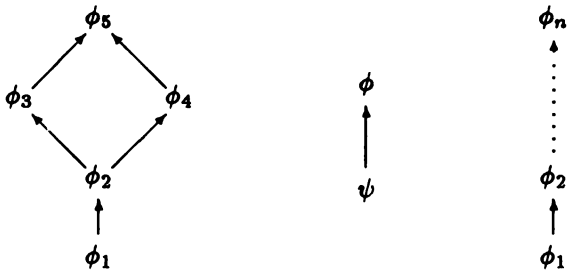


Figure 1: Three OTPs

of defaults—those below in the ordering override those above, if there is a conflict. Sentences can be a default relative to one sentence but not relative to another. We can specify which sentences are to override exactly which others.

This way of presenting a theory we call an *ordered theory presentation*, or OTP for short. Consider the first example of an OTP in figure 1 to make the above discussion of priority a little more concrete. In that ordered presentation, there is one 'fact', namely ϕ_1. It has a greater priority than the other sentences. The others are defaults, but still, some are stronger than others. For example, ϕ_3 is stronger than ϕ_5, weaker than ϕ_2 and incomparable in strength with ϕ_4. Thus, the arrow is read as 'is stronger than' or 'dominates'. This information is part of the knowledge being represented.

If a sentence dominates another, that means that it can override it if the two conflict. The meaning of the second OTP in figure 1 is $\phi \wedge \psi$ if ϕ and ψ are mutually consistent; otherwise it is ψ with as much of ϕ as is consistent with ψ. Thus, if they are inconsistent, ψ overrides ϕ. But this overriding, when it happens, is in general only partial. ψ doesn't override all of ϕ, just those bits which conflict with it. The machinery needed for this is described later in the paper.

OTPs were first described in [Rya91], where they were called 'structured theories'. This paper is self contained, but some technical details and many proofs have been omitted here to leave space for new results. The most complete account of OTPs to date is [Rya92a], copies of which are available from the author.

The remainder of the paper is organised as follows. In §2 we give examples of ordered theory presentations and define their semantics. In §3 we examine some standard examples of default reasoning using OTPs, and in §4 we show the relation with other default systems. Finally, in §5 we show how to use OTPs for belief revision.

2 Ordered theory presentations

An ordered presentation of a theory is a *partially ordered multi-set of sentences*. Sentences lower in the ordering take priority over those above. Earlier we simplified by saying that it was a partially ordered *set*, but we have to consider multi-sets because the same sentence may occur twice, in different places in the order. An informal syntax of graphs for OTPs was used in §1, which is used for much of the paper; a more formal notation is introduced in §2.2.

2.1 Examples and motivation

This section is intended to illustrate by example the intended behaviour of OTPs. The reader can check the examples against his or her intuitions. All of them work out successfully in the formalism described in the paper. While reading these examples, it is important to keep the following points in mind:

1. In an OTP, sentences lower in the ordering take precedence over those above.
2. When a sentence lower in the ordering contradicts a sentence above it in the ordering, the lower sentence overrides the higher one. But in general, this overriding is only partial. The lower sentence does not cancel the effect of the higher one completely.
3. In evaluating an OTP (that is, in working out the theory it presents), the idea is to use as much of the available information as possible but to avoid contradictions.

We take the underlying logic (classical propositional logic or classical predicate logic) as given.

Example 2.1 Here are two OTPs and the ordinary sentences to which they are equivalent.

$$\begin{array}{c} p \wedge q \\ \uparrow \\ \neg p \end{array} \quad \equiv \quad \neg p \wedge q \qquad\qquad \begin{array}{c} p \wedge q \\ \uparrow \\ \neg p \vee \neg q \end{array} \quad \equiv \quad p \leftrightarrow \neg q.$$

In the first case, the OTP consists of the sentences $\neg p$ and $p \wedge q$, but with the former overriding the latter. Thus, $p \wedge q$ is a default relative to $\neg p$. The OTP means that we want $\neg p$ first and foremost, and subject to that, as much of $p \wedge q$ as possible. But $p \wedge q$ conflicts with $\neg p$, so we can't have it all; we can only have the q component. Therefore we get $\neg p$ and q.

In the second case, the default $(p \wedge q)$ is the negation of the given sentence $(\neg p \vee \neg q)$. The overall effect of the OTP is to give us the certain sentence (the $\neg p \vee \neg q$), and then as much of the default as is consistent. Of $p \wedge q$, we can have either p or q but not both. That is why we end up with $p \leftrightarrow \neg q$.

Example 2.2

$$\begin{array}{c} p \\ \uparrow \\ q \\ \uparrow \\ \neg p \vee \neg q \end{array} \quad \equiv \quad \neg p \wedge q$$

This is like the second case of example 2.1, except now there is a priority expressed between p and q. This priority is expressed by their location in the ordering. The bottom sentence (the most important) says that we want one of p and q to fail; but subject to that we want q. This gives us $\neg p \wedge q$, since they are consistent. Then, subject to all *that*, we want p. But we've ruled that out by now, so we end up with $\neg p \wedge q$.

Example 2.3 This example will turn out to have importance in §5.

$$\begin{array}{c} p \wedge q \wedge r \\ \uparrow \\ \neg p \vee \neg q \vee \neg r \\ \uparrow \\ (p \leftrightarrow q) \vee \neg r \end{array} \quad \equiv \quad (p \leftrightarrow q) \wedge (p \leftrightarrow \neg r)$$

To see this is correct, separate the cases of r and $\neg r$. If r, then we must have $p \leftrightarrow q$ in order to satisfy the most important sentence (the bottom one). To satisfy the next sentence, we must have $\neg p$ or $\neg q$. Since we already have $p \leftrightarrow q$, this means we have $\neg p \wedge \neg q$. Now we have determined the value of all three atoms, for we have $\neg p \wedge \neg q \wedge r$. On the other hand, if $\neg r$ then both the bottom sentence and the middle one are satisfied. We want as much of the top one as possible, which is $p \wedge q$. Therefore, we get $p \wedge q \wedge \neg r$. The presentation is thus equivalent to $(\neg p \wedge \neg q \wedge r) \vee (p \wedge q \wedge \neg r)$, which is elementarily equivalent to $(p \leftrightarrow q) \wedge (p \leftrightarrow \neg r)$.

Example 2.4

$$\begin{array}{c} \forall x.\, p(x) \\ \uparrow \\ \exists x.\, \neg p(x) \end{array} \quad \equiv \quad \exists x.\, (\neg p(x) \wedge \forall y.\, (x \neq y \rightarrow p(y)))$$

The more important sentence (the bottom one) says that there is at least one individual which has not got the property p. But, subject to satisfying that, we want to satisfy as much of the upper sentence as possible; it says that all individuals have the property p. We conclude therefore, that precisely one individual fails p; all the others satisfy it. This is stated by the theory on the right.

The examples illustrate the intended behaviour of ordered presentations. Our aim in the next section is to define their semantics formally. We do so in a logically clean way, so that our definitions do not interfere with the mechanism of the underlying logic.

2.2 The semantics of ordered theory presentations

We will define the models of ordered theory presentations. Since the sentences of an OTP are in general inconsistent, we cannot expect its models to satisfy all the sentences. Instead, they should satisfy the lower ones, and then as much of the higher ones as possible. To achieve this we define for each OTP an ordering of the interpretations of the language. This ordering ranks interpretations according to how well they satisfy the sentences of the OTP; and this ranking respects the ordering of the sentences in the OTP. Then models of the OTP are taken to be the maximal interpretations. This strategy of ordering models is well-known in the default reasoning literature [Bes88, McC80, Sho88]

First, it is necessary to have a more formal notation for OTPs than the graphs of the last section. We have seen that an ordered theory presentation is a collection of sentences equipped with a partial order. But to cover the case that the same sentence occurs several times in different places in the presentation, it is necessary to posit a 'carrier set' on which the order is defined and whose points are labelled by sentences.

Definition 2.5 An ordered theory presentation Γ is a tuple $\langle X, \leqslant, F \rangle$ where X is a finite set (the carrier set), $\leqslant$ is a partial order on X, and F is a function mapping X to sentences.

The intuitive meaning of the ordering is: if $x < y$ then the sentence $F(x)$ has greater priority (or more influence) than $F(y)$. This information is used when $F(x)$ and $F(y)$ conflict. We will assume that we are working with a fixed language L over propositional logic or predicate logic with equality[1], with interpretations $\mathcal{M}$ and a satisfaction relation $\Vdash \subseteq \mathcal{M} \times L$ between interpretations of the language and sentences.

As already stated, to define the models of an ordered theory presentations Γ we define an ordering $\sqsubseteq^{\Gamma}$ on interpretations in $\mathcal{M}$ which measures how well an interpretation satisfies Γ. $M \sqsubseteq^{\Gamma} N$ shall mean that N is as good (or better) than M at satisfying Γ. Models of Γ are then taken to be maximal interpretations in this ordering[2]. The definition of the ordering relies on orderings $\sqsubseteq_{\phi}$, one for each sentence ϕ of the language.

[1] In fact, the definitions and results presented here work with other logics, including modal and intuitionistic logics. But in this paper we restrict ourselves to classical logic.

[2] The technique of ordering interpretations which is used in in this paper is well-established in the literature. It originates in McCarthy's first circumscription paper [McC80], and has been generalised in various ways [Sho88, Bes88, KLM90, Vel91, etc.]. In all of those papers, the ordering works in the opposite way to the one we have used for OTPs, that is, $M < N$ means M is better than N; and therefore, one is interested in minimal models. The reader may wonder why we chose to fly in the face of

The relation $\sqsubseteq_\phi$ grades interpretations according to how well they satisfy ϕ. To define $\sqsubseteq_\phi$, it is necessary to define a notion which we call 'natural entailment', written $\models\!\!\!=$. This definition in turn relies on the notion of the *monotonicities* of a sentence. We start therefore with the definition of monotonicities. Then we proceed to the definition of $\models\!\!\!=$, then $\sqsubseteq_\phi$, then $\sqsubseteq^\Gamma$.

The positive monotonicities of a sentence are the predicates whose extension can be increased in any model of the sentence. The negative monotonicities are the predicates which can be decreased in the model. We define this formally as follows:

Definition 2.6 1. In predicate logic, the *extension* of a predicate symbol p in a model is the set of tuples of which p is true in the model. In propositional logic, the extension of a proposition p in a model is a singleton $\{*\}$ if p is true in the model; if p is false, it is $\varnothing$.

2. Extensions are naturally ordered by inclusion. We define $M \leqslant^p N$ if M and N are exactly alike except that the p-extension of M is included in that of N.

3. If $\phi \neq \bot$ then ϕ is *monotonic in* p (written $p \in \phi^+$) if $M \leqslant^p N$ and $M \Vdash \phi$ imply $N \Vdash \phi$. Similarly ϕ is *anti-monotonic in* p ($p \in \phi^-$) if $N \leqslant^p M$ and $M \Vdash \phi$ imply $N \Vdash \phi$. The case that $\phi = \bot$ is handled separately, for technical reasons which will become clear; we define $\bot^+ = \bot^- = \varnothing$.

That is to say, $p \in \phi^{+(-)}$ if increasing (decreasing) the extension of p in any model of ϕ results in another model of ϕ.

Example 2.7 Let $(L, \mathcal{M})$ be classical propositional logic over $\{p, q\}$. For several examples of ϕ, the sets ϕ^+ and ϕ^- are shown in the following table.

ϕ	ϕ^+	ϕ^-
$\top$	$\{p,q\}$	$\{p,q\}$
p	$\{p,q\}$	$\{q\}$
q	$\{p,q\}$	$\{p\}$
$p \wedge q,\ p \vee q$	$\{p,q\}$	$\varnothing$
$p \rightarrow q$	$\{q\}$	$\{p\}$
$p \leftrightarrow q$	$\varnothing$	$\varnothing$
$\bot$	$\varnothing$	$\varnothing$

Example 2.8 Let $(L, \mathcal{M})$ be classical predicate logic over p (unary) and q (binary).

ϕ	ϕ^+	ϕ^-
$\forall x.\, p(x)$	$\{p,q\}$	$\{q\}$
$\exists x.\, p(x)$	$\{p,q\}$	$\{q\}$
$\forall x.\, \exists y.\, q(x,y)$	$\{p,q\}$	$\{p\}$
$\forall x.\, (p(x) \rightarrow \exists y.\, q(x,y))$	$\{q\}$	$\{p\}$
$\forall x.\, \forall y.\, (q(x,y) \rightarrow q(y,z))$	$\{p\}$	$\{p\}$

In classical logic we may characterise $\phi^\pm$ more syntactically, by means of positive and negative occurrences. Recall that p *occurs positively in* ϕ if it occurs in ϕ within the scope of an even number of negation operators, after the operators $\rightarrow$ and $\leftrightarrow$ have been unpacked in terms of their standard definitions. Similarly, if in such circumstances it appears in the scope of an odd number of negation signs then it *occurs negatively*.

Proposition 2.9 $p \in \phi^{+(-)}$ iff ϕ can be written with only positive (negative) occurrences of p.

Notice that the definition is *semantic* in the sense that it is not sensitive to the way ϕ is written. That is, writing $\phi \models\!\!\!=\!\!\models \psi$ if $\phi \models \psi$ and $\psi \models \phi$, we have that $\phi \models\!\!\!=\!\!\models \psi$ implies $\phi^\pm = \psi^\pm$.

Having defined monotonicities, we turn to the definition of natural entailment. Let ϕ and ψ be sentences of L.

Definition 2.10 ϕ *naturally entails* ψ, written $\phi \models\!\!\!= \psi$, if $\phi \models \psi$, and $\phi^+ \subseteq \psi^+$, and $\phi^- \subseteq \psi^-$.

Natural entailment is a sub-relation of ordinary entailment; in addition to ordinary entailment we require that the monotonicities of the premise be preserved by the conclusion.

Remark 2.11 1. $\models\!\!\!=$ is a reflexive and transitive relation.

2. We have that $p \wedge q \models\!\!\!= p$ and $p \wedge q \models\!\!\!= p \vee q$, but $p \wedge q \not\models\!\!\!= p \leftrightarrow q$ and $p \not\models\!\!\!= p \vee q$. Moreover, $\bot \models\!\!\!= \phi$ for all ϕ. (That was the reason for requiring $\bot^\pm = \varnothing$.) The full picture for natural and ordinary entailment for propositional logic with the predicates p, q is given in figure 2.

this well-established convention, in choosing to order interpretations in the opposite sense and therefore to seek $\sqsubseteq^\Gamma$-*maximal* interpretations. There are two reasons. The first is that the fact that other workers order models in the opposite way is for the historic reason that in circumscription one wants to minimise abnormality predicates; this reason does not apply in the more abstract setting of OTPs. On the contrary, it is more intuitive to move *upwards* in an ordering when one is moving to better and better models. The second reason is that one typically looks at *ascending chains* and *maximal elements* in domain theory and information systems theory, where we see links with our work. Cf. proposition 2.19.

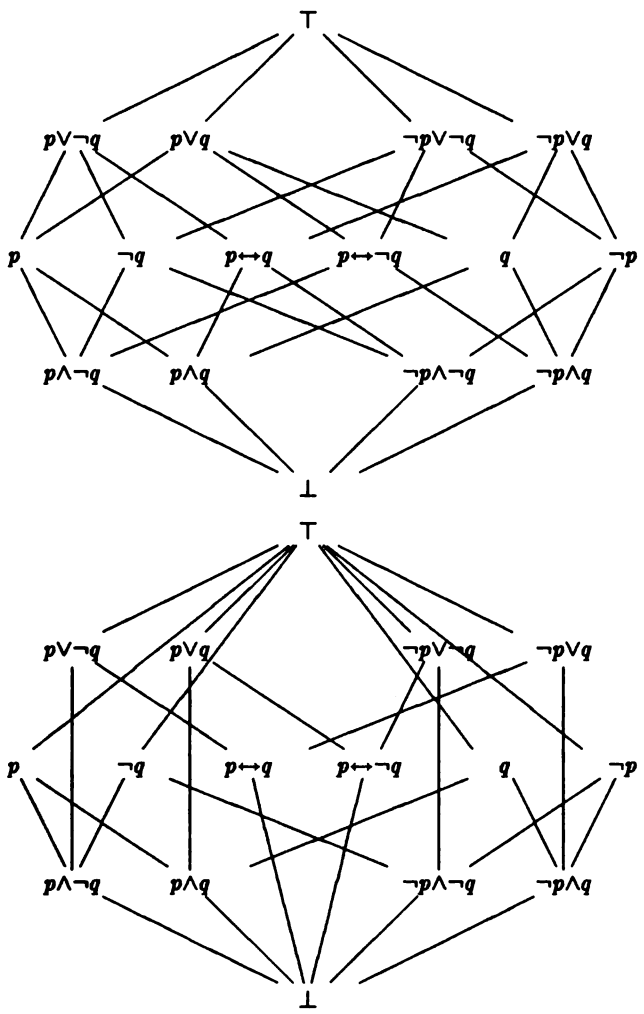


Figure 2: The *ordinary* and *natural* consequence relations over $\{p, q\}$

3. Also, $\forall x.\, p(x) \models \psi$ implies ψ can be written with no negative occurrences of p and no occurrences of any other predicate.

Natural entailment is something like 'relevant entailment'; it stops us adding irrelevant disjuncts in our conclusions. The simplicity of the definition and the fact that it is based on satisfaction by models ensures that there is nothing untoward going on. In particular, if ϕ and ψ are classically equivalent then they are naturally equivalent; indeed

$$\phi \mid\!\models \psi \text{ iff } \phi \models\!\mid \psi.$$

Our interest in natural entailment is in order to achieve the definition of $\sqsubseteq_\phi$, to which we now turn. As stated, $M \sqsubseteq_\phi N$ means that N is as good at satisfying ϕ as M is. It is not just that N satisfies ϕ and M does not; perhaps neither satisfy ϕ, but N more nearly does. For example, let M be a propositional interpretation which assigns false to both p and q; and let N assign true and false to p and q respectively. Then $M \sqsubseteq_{p\wedge q} N$, while $N \not\sqsubseteq_{p\wedge q} M$. Neither satisfy $p \wedge q$, but at least N satisfies p; M doesn't satisfy either of p and q.

This example shows that one has to look at which *consequences* of ϕ are satisfied by M and N. However, defining $M \sqsubseteq_\phi N$ to mean that N satisfies all the consequences of ϕ which M does gives us precisely the bipartite ordering rejected in the preceding paragraph. This is because ϕ has too many irrelevant consequences; we should just look at the *natural* ones.

Definition 2.12 M satisfies ϕ no worse than N, written $M \sqsubseteq_\phi N$, if for each ψ such that $\phi \models\!\mid \psi$, $M \Vdash \psi$ implies $N \Vdash \psi$.

Examples will be given shortly. It is easy to verify that

Proposition 2.13 1. $\sqsubseteq_\phi$ is a pre-order, that is to say, it is reflexive and transitive.

2. If $\phi \neq \bot$, the maximal elements of $\sqsubseteq_\phi$ (which are in fact maxi*mum*) are just the models of ϕ.

3. If $\phi \mid\!\models \psi$ then $\sqsubseteq_\phi = \sqsubseteq_\psi$.

We have defined, for each sentence ϕ, an ordering on interpretations $\sqsubseteq_\phi$ which measures the extent to which interpretations satisfy ϕ. If M satisfies ϕ to the fullest extent (that is, if it simply satisfies it) then M is $\sqsubseteq_\phi$-maximum. If M does not fully satisfy ϕ then it may satisfy it to a greater, lesser, equal or incomparable extent than some N which perhaps also fails fully to satisfy ϕ. We now define $\sqsubseteq^\Gamma$ in terms of $\sqsubseteq_\phi$ as follows.

Definition 2.14 $M \sqsubseteq^\Gamma N$ if for each $x \in X$, $M \not\sqsubseteq_{F(x)} N$ implies there exists $y \leqslant x$ such that $M \sqsubset_{F(y)} N$.

One can read this as saying: N is as good as M overall $[M \sqsubseteq^\Gamma N]$ if whenever it appears not to be so at a point x $[M \not\sqsubseteq_x N]$ then there is a more important point y $[y \leqslant x]$ where N is doing better than M $[M \sqsubset_y N]$. Informally, the definition says: if things appear to go wrong at a particular x, then they go well at some y in a more important position than x.

Remark 2.15 The definition of $\sqsubseteq^\Gamma$ is more perspicuous if Γ is linear. Let Γ be the third OTP of figure 1. Then $\sqsubseteq^\Gamma$ is the lexicographic combination of the $\sqsubseteq_{\phi_i}$s, i.e.

$$\begin{aligned} M \sqsubseteq^\Gamma N \quad \text{iff} \quad & M \sqsubset_{\phi_1} N \\ & \text{or } (M \sqsubseteq_{\phi_1} N \text{ and } M \sqsubset_{\phi_2} N) \\ & \text{or } (M \sqsubseteq_{\phi_1,\phi_2} N \text{ and } M \sqsubset_{\phi_3} N) \\ & \text{or } \ldots \text{ or} \\ & (M \sqsubseteq_{\phi_1 \ldots \phi_{n-1}} N \text{ and } M \sqsubset_{\phi_n} N). \end{aligned}$$

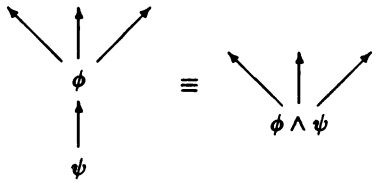
Proposition 2.16 $\sqsubseteq^\Gamma$ is also a pre-order.

Finally, we define the models of Γ. They are simply the interpretations which are rated maximally by $\sqsubseteq^\Gamma$.

Definition 2.17 $M \Vdash \Gamma$ if M is $\sqsubseteq^\Gamma$-maximal.

These definitions represent the guts of the system we propose. Before turning to examples, we present some results.

Proposition 2.18 If ϕ and ψ are mutually consistent then



Proposition 2.19 Let Γ be an OTP and $M \in \mathcal{M}$. There exists $N \in \mathcal{M}$ such that $M \sqsubseteq^\Gamma N$ and N is $\sqsubseteq^\Gamma$-maximal.

Proposition 2.20 If $\Gamma \models \phi$ then $\phi \neq \bot$.

The last of these says that no contradictions may be concluded from any OTP. This may seem surprising, but is really quite rational!

The proofs of these propositions may be found in [Rya92a]. They rely on the compactness of the underlying logic (which in this paper we have assumed is classical propositional or predicate logic).

We now turn to some examples, applying the definitions given so far.

Example 2.21 The working for example 2.3 is given in figure 3. For each sentence ϕ in the OTP, the ordering $\sqsubseteq_\phi$ is shown. Then these are combined in the manner of remark 2.15 to yield the final model ordering, whose maximal elements are 001 and 110. The formula with precisely these models is $(p \leftrightarrow q) \wedge (p \leftrightarrow \neg r)$.

In figure 3, we show the ordering on interpretations by means of similar diagrams to the ordering of sentences in OTPs. It is hoped that this is not confusing. If such a diagram has sentences at its nodes, it is an OTP. If it has interpretations at its nodes, it is the diagram corresponding to an ordering $\sqsubseteq_\phi$ or $\sqsubseteq^\Gamma$ for some sentence ϕ or OTP Γ.

Further examples of $\sqsubseteq_\phi$ and $\sqsubseteq^\Gamma$ for propositional and predicate logic are given in [Rya92a].

3 Representing defaults in OTPs

We will concentrate on two classic examples, one about inheritance and one about temporal reasoning. To the reader acquainted with default systems they will be very familiar. Although hackneyed, they are excellent examples for showing the key differences between formalisms.

Inheritance example. We will consider the well-known example concerning birds and penguins and whether they can fly. The class of penguins is a subclass of the class of birds. But the property of being able to fly, which holds of birds by default, is not inherited by penguins. In the usual formulation of this example, we have the *factual* premise 'Penguins are birds', together with the *defaults* 'Birds can fly' and 'Penguins cannot fly'. Using predicates to represent the obvious classes, we have $\forall x.\,(p(x) \rightarrow b(x))$, $\forall x.\,(b(x) \rightarrow f(x))$, and $\forall x.\,(p(x) \rightarrow \neg f(x))$.

We want the following results:

1. If Fred is stated to be a bird (whether he is also a penguin or not is not stated), we want to conclude that he can fly.
2. But if it is stated that he is a penguin, we want to conclude that he cannot fly.

The reason this example is interesting is that there are two defaults which compete in certain circumstances. It is easy to get result 1 correctly, but it is in the case of result 2 that the defaults conflict. Our intuition that the second of the two defaults should have priority and block the application of the first is based on the *specificity principle*, which states that *defaults about a specific class of objects take priority over defaults about a more general class.* We use this principle to order the sentences, obtaining for case 1:

$$\begin{array}{c} \forall x.\,(b(x) \rightarrow f(x)) \\ \uparrow \\ \forall x.\,(p(x) \rightarrow \neg f(x)) \\ \uparrow \\ \forall x.\,(p(x) \rightarrow b(x)) \\ \wedge\, b(\text{Fred}) \end{array}$$

This OTP proves f(Fred) as required. The OTP for case 2 of the example has p(Fred) instead of b(Fred), and proves $\neg f$(Fred).

This shows the fundamental difference between the two approaches to default representation discussed in the Introduction. We have here a set of sentences which we wish to represent, but like much of what is taken to be common knowledge, they conflict with each other. To handle this, we regard some as being weaker than others. The stronger sentences may partially override the weaker ones. These are the basic principles of this second way of representing defaults.

Multiple inheritance The framework of ordered theory presentation is much better suited to inheritance examples than the above analysis indicates. Instead of expressing the fact that penguins are birds by the formula $\forall x.\,(p(x) \rightarrow b(x))$, we construct a specification for birds, and then construct a specification for penguins by stating that they inherit the properties of birds.

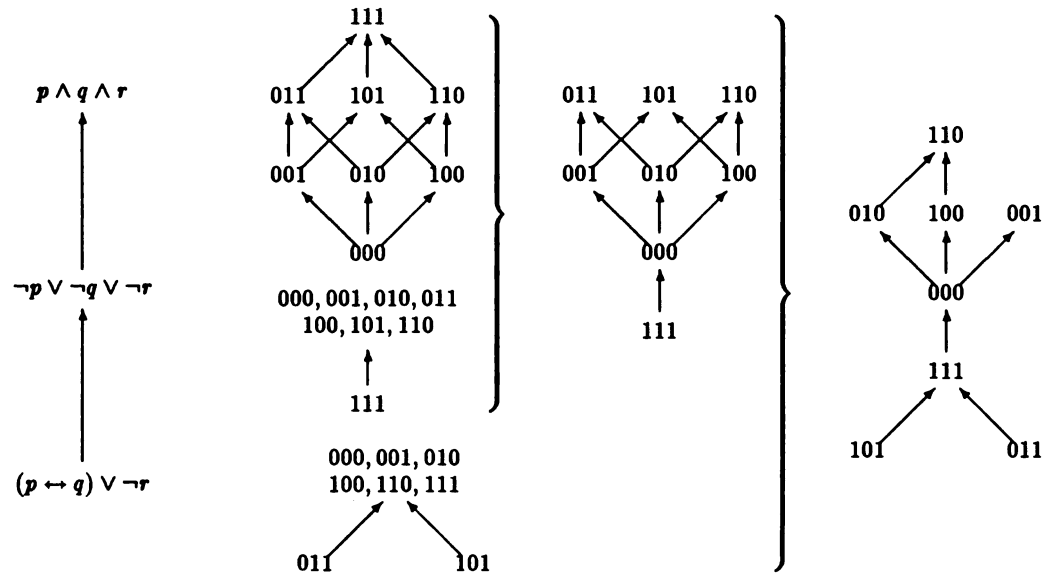


Figure 3: The working for example 2.21

Our specification for birds is

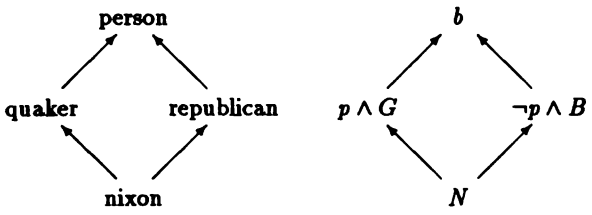
$$f \wedge e$$

meaning that they *f*ly and lay *e*ggs. Now we construct the specification of penguins by inheriting the properties of birds and overriding as necessary:

$$f \wedge e$$
$$\uparrow$$
$$\neg f$$

The ordering for the OTP comes straight from the inheritance ordering.

Although our examples of OTPs in this paper have been linear, the definitions allow the sentence ordering to be partial. (Examples of partially ordered theory presentations are given in [Rya92a].) In that way we can handle multiple inheritance. For example, a rendering of the familiar inheritance situation for Nixon is given in the first of the following diagrams (in which the arrows mean 'is-a'), while the second shows the corresponding OTP, in which we have propositions expressing the properties *b*iped, *p*acifist, believer in *G*od, supporter of *B*ush, and being named *N*ixon.



Temporal example. We will describe our solution to the Yale Shooting problem. We assume familiarity with the description of the problem and with the well-known pitfalls in representing the facts and defaults in a way which yields the correct prediction [HM86, Kau86, Sho88, Bak91, Sha92].

At the basis of the problem there are two competing defaults, one expressing the persistence of the loadedness of the gun and the other the persistence of the man's aliveness. The essence of a large class of solutions focusses on the idea, due to Y. Shoham [Sho88], that defaults relating to earlier states of the system should take priority over defaults relating to later states. Thus, we may stipulate a principle for persistence defaults[3], analogous to the specificity principle for inheritance defaults. The *chronology principle* states that: *defaults about an earlier state take priority over defaults about a later state.*

We can use this principle directly in the context of ordered theory presentations. To illustrate this, we shall dramatically simplify the problem by coding it in propositional logic with three states, which we represent by indices on the propositions[4].

[3]It is important to note that this principle is appropriate only when using defaults to predict the outcome of action sequences, i.e. for 'prediction problems'. It is not appropriate for other examples of uses of persistence defaults, such as 'explanation problems' where it is desired to account for a known outcome, in which this principle manifestly gets the wrong answer. An example of this is H. Kautz' 'stolen car problem' [Kau86].

[4]This is not the coding of the example given in Reiter's logic by Hanks and McDermott in the usual paper.

Let the three states be represented by the set $\{1,2,3\}$, in which 1 is the result of the loading action, 2 is the result of waiting and 3 results from the shoot action. Let ℓ_i and a_i mean respectively that the gun is loaded and the man is alive in state i. We have the facts

$$\ell_1 \qquad a_1 \qquad \ell_2 \rightarrow \neg a_3$$

and we have the defaults

$$a_i \rightarrow a_{i+1} \qquad \ell_i \rightarrow \ell_{i+1} \qquad (i \in \{1,2\})$$

Notice that we have not represented the fact that being loaded in a state is an exception to the persistence of alive in the state which follows a shoot action—as was done in the original coding of [HM86]. We do not need to do this, because that fact is represented by the chronology principle, which says that the persistence of earlier fluents shall have priority over the persistence of later ones. We use this to arrive at the ordered theory presentation:

$$\begin{array}{c} a_2 \rightarrow a_3 \\ \uparrow \\ (\ell_1 \rightarrow \ell_2) \wedge \\ (a_1 \rightarrow a_2) \\ \uparrow \\ \ell_1 \wedge a_1 \wedge \\ (\ell_2 \rightarrow \neg a_3) \end{array}$$

This OTP proves $\neg a_3$ as required.

We do not intend to conclude from this analysis that the logic of ordered theory presentations is superior to all the other default systems because it obtains the correct answer to the Yale Shooting Problem. Such a conclusion would be naïve for many reasons. For example, our solution is a crude application of the chronology principle, but, as H. Kautz' stolen car example shows [Kau86], this is not appropriate for all examples of reasoning about actions. We have illustrated that the theory of OTP given in this paper does correctly implement prioritisation of defaults in a natural way which allows for clear knowledge representation. We also hope that we have shown that the representation of defaults, and interacting defaults in particular, is clearer in the theory of OTPs than in many of its rivals.

We have simplified rather dramatically by using a propositional language and making explicit the identities of the states. This simplification is justified since the same problem occurs in this simpler setting as occurred in Hanks and McDermott's, but the simpler setting is rather easier to understand. However, it is true that the simpler setting may not do justice to some of the subtler solutions to the problem which have appeared in the literature. As these are not the main interest of this paper, I feel this is not a significant loss.

4 Relation with other default systems

4.1 Circumscription

PRELIMINARY REPORT

The aim of this topic, which has not yet been achieved, is to provide theorems which show how to translate between OTPs and circumscriptive theories. In this section we give the story so far by means of results, examples and conjectures.

We assume familiarity with the ideas of circumscription [McC80, Lif85], prioritised circumscription [Lif87], and also with *propositional* circumscription. Circumscribing a proposition in a theory means trying to make it false, just as circumscribing a predicate means trying to make its extension as small as possible. It is easy to show that the circumscription of a set of propositions (allowing another set to vary) in a propositional theory is again a propositional theory.

We also adopt the following notation:

- $\mathrm{Circ}_{p;z}(\phi)$ is the circumscription of p in ϕ, allowing z to vary and keeping everything else constant. p and z may be tuples of propositions or predicates.
- $\mathrm{Circ}_p^z(\phi)$ is also the circumscription of p in ϕ, but keeping z fixed and allowing everything else to vary.

It turns out that OTPs translate into circumscriptive theories in which everything is allowed to vary, so we will often be interested in the special case $\mathrm{Circ}_p^{\varnothing}(\phi)$, which we abbreviate to $\mathrm{Circ}_p(\phi)$.

From circumscription to OTPs. The simplest case is $\mathrm{Circ}_p(\phi)$ where p is a single proposition or predicate. The corresponding OTPs are respectively

$$\begin{array}{cc} \neg p & \forall \underline{x}.\neg p(\underline{x}) \\ \uparrow & \uparrow \\ \phi & \phi \end{array}$$

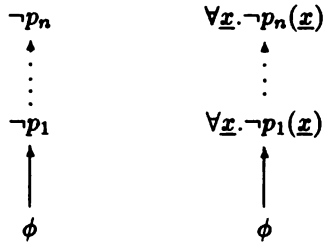
This follows from the facts that:

- If p is a proposition, $M \sqsubseteq_{\neg p} N$ iff $N \Vdash p$ implies $M \Vdash p$.
- If p is a predicate, $M \sqsubseteq_{\forall \underline{x}.\neg p(\underline{x})} N$ iff M, N are isomorphic structures in terms of the functions of the language and (modulo that isomorphism) the p-extension of N is included in that of M.

The parallel circumscription of several propositions or predicates $\mathrm{Circ}_{p_1 \ldots p_n}(\phi)$ is respectively

$$\begin{array}{ccccccc} \neg p_1 & \ldots & \neg p_n & \forall \underline{x}.\neg p_1(\underline{x}) & \ldots & \forall \underline{x}.\neg p_n(\underline{x}) \\ \nwarrow & & \nearrow & \nwarrow & & \nearrow \\ & \phi & & & \phi & \end{array}$$

and the prioritised circumscription $\mathrm{Circ}_{p_1>\ldots>p_n}(\phi)$ becomes



since, as implied by [Lif87, eq. 9], in the context of the given priorities we have

$$\begin{aligned} p \sqsubset p' \quad \text{iff} \quad & p_1 \sqsubset p'_1 \\ & \text{or } (p_1 \sqsubseteq p'_1 \text{ and } p_2 \sqsubset p'_2) \\ & \text{or } (p_{1,2} \sqsubseteq p'_{1,2} \text{ and } p_3 \sqsubset p'_3) \\ & \text{or } \ldots \text{ or} \\ & (p_{1\ldots n-1} \sqsubseteq p'_{1\ldots n-1} \text{ and } p_n \sqsubset p'_n). \end{aligned}$$

Compare remark 2.15. Here, $p_2 < p_1$ means that p_1 is circumscribed with a greater priority than p_2, while $p \sqsubseteq p'$ ($p \sqsubset p'$) means that the extension of p is (strictly) included in that of p'.

A corollary of this analysis is the obvious definition of 'partially prioritised circumscription', in which predicates are circumscribed with priorities specified by a partial order.

Definition 4.1 Let $<$ be a partial order on the tuple p of predicates $p_1, \ldots, p_n$. Then $p \sqsubseteq p'$ if for each i with $p_i \sqsubseteq p'_i$ there exists j with $p_i \leqslant p_j$ such that $p_j \sqsubset p'_j$.

Compare definition 2.14.

From OTPs to Circumscription. The situation in this direction is rather more complicated. We will consider only the simplest case of

It will not be hard to generalise. Let Γ be this OTP. The basic idea is to split $\phi(x)$ into 'components' $\phi_1(x) \wedge \ldots \wedge \phi_n(x)$ and consider the circumscription

$$\mathsf{Circ}_{\mathsf{ab}_1 \ldots \mathsf{ab}_n}(\psi \wedge \bigwedge_i (\neg\phi_i(x) \rightarrow \mathsf{ab}_i(x)))$$

But what constraints should there be on the way ϕ is split into components? As a minimum one would expect to require *conjunctive normal form*, but this is not good enough, as we now show.

Example 4.2 Suppose $\phi = p \wedge q$. There are several normal forms, two such being $p \wedge q$ itself and $p \wedge (\neg p \vee q)$. We have

$$\Gamma \equiv \mathsf{Circ}_{\mathsf{ab}_1,\mathsf{ab}_2}(\psi \wedge (p \rightarrow \mathsf{ab}_1) \wedge (q \rightarrow \mathsf{ab}_2))$$

whatever ψ may be, but we also have that

$$\Gamma \not\equiv \mathsf{Circ}_{\mathsf{ab}_1,\mathsf{ab}_2}(\psi \wedge (p \rightarrow \mathsf{ab}_1) \wedge (\neg p \vee q \rightarrow \mathsf{ab}_2)).$$

For example, set $\psi = \neg p$. Then $\Gamma \equiv \neg p \wedge q$, but the right-hand side is simply $\neg p$.

The example shows that we probably also require that the split of ϕ must not contain unnecessary occurrences of propositions or predicates of the 'wrong' polarity. It is a consequence of Lyndon's Theorem [CK90] that:

Proposition 4.3 For any formula ϕ there is an equivalent formula ϕ' such that every predicate occurring positively (negatively) in ϕ' occurs positively (negatively) in every formula equivalent to ϕ.

In other words, we can find for every ϕ an equivalent ϕ' with no eliminable occurrences of predicates. We conjecture that this is the normal form we seek.

4.2 Formal properties of OTPs

The study of default systems has been transformed by a new concern, namely the formal properties of the generated consequence relation. The first default systems introduced in the 1980 special issue of *Artificial Intelligence* [AIJ80] did not even have well-defined consequence relations. Gabbay and Clark [Gab91, CG88] first observed that, instead of focussing on the *negative* properties of such consequence relations, that is, their *non*-monotonicity, one should instead ask what positive properties they have. They gave the name 'cautious monotonicity' to the property

$$\frac{\Phi \vdash \phi \quad \Phi \vdash \psi}{\Phi, \phi \vdash \psi}.$$

This property, which is weaker than full monotonicity, has become widely accepted as a desirable property for default systems.

The story of the properties of default consequence relations has been pursued in the work of Kraus/Lehmann/Magidor [KLM90, Leh89] and also by Makinson [Mak88, Mak92]. Makinson's [Mak92] is, in my opinion, the most authoritative and systematic study to date. He describes and motivates a set of conditions on a default consequence relation and analyses existing systems according to whether they have the conditions. In this section we outline his principal conditions and check the theory of OTPs of this paper against them.

In Makinson's work, the expression $\Phi \mathrel{|\!\sim} \psi$ should be read as: ψ follows from Φ *in the context of an understood set of defaults*. It is unfortunate (and detracts slightly from Makinson's systematic study) that these defaults are nowhere made explicit. Consequently, the behaviour of the consequence relation under variations

of the defaults—and for that matter, questions of default representation—are not examined at all in his work.

Makinson's conditions also refer to classical consequence, written $\vdash$. $\Phi \vdash \psi$ is to be read as ψ follows from Φ without using the defaults. The understood set of defaults can be thought of as augmenting classical consequence to default consequence. Therefore, the first property we may expect is

Supraclassicality:

$$\frac{\Phi \vdash \psi}{\Phi \mid\sim \psi}.$$

It says that anything which can be derived without the defaults can also be derived with them.

The next three conditions are together called '**cumulativity**'. The first is simply

Inclusion: if $\psi \in \Phi$ then $\Phi \mid\sim \psi$.

The next two are weak forms of the standard Tarski conditions of cut and monotonicity:

Cautious monotonicity:

$$\frac{\Phi \mid\sim \phi, \text{ for all } \phi \in \Psi \quad \Phi \mid\sim \psi}{\Phi, \Psi \mid\sim \psi}$$

Weak cut:

$$\frac{\Phi \mid\sim \phi, \text{ for all } \phi \in \Psi \quad \Phi, \Psi \mid\sim \psi}{\Phi \mid\sim \psi}$$

For the justification of these principles in intuitive terms, we cannot do better than quote Makinson. "Cut may be seen as expressing a determination not to allow the length, intricacy or manner of a derivation of a conclusion to reduce the freedom with which it is used in further inference. There is no 'diminution of usability' with respect to distance from origins. Once inferred, a proposition may be called upon in conjunction with the original information, unless genuinely new (i.e. uninferable) information is also added. Cautious monotonicity, on the other hand, may be seen as expressing a certain irreversibility in the drawing of conclusions. Once inferred, a proposition may be retained irrespective of what other inferred propositions are added to the stock of usable information. We need never go back unless, once more, genuinely new information is brought in" [Mak92].

The next condition we will consider is

Distributivity: If Φ and Ψ are $\vdash$-closed sets of sentences (that is, $\Phi \vdash \phi$ implies $\phi \in \Phi$, and similarly for Ψ) then

$$\frac{\Phi \mid\sim \phi \quad \Psi \mid\sim \phi}{\Phi \cap \Psi \mid\sim \phi}.$$

Makinson considers other conditions, but these are the principal ones.

We have already noted that Makinson's conditions make no reference to the set of defaults which are implicit in the relation $\mid\sim$. On the other hand, one of the attractive features of the framework of Ordered Theory Presentations as a default system is that there is *no difference* between defaults on one hand and 'sure facts' or facts on the other, except the priority they are given in the ordering. We view this as a desirable feature since we believe that, philosophically, the so-called sure facts and the defaults have the same provenance. They should all form part of the theory from which we make deductions. A sentence does not have the status of a default in isolation, but only in relation to other sentences; to be precise, it is a default relative to those sentences which can override it.

Nevertheless, we can go quite some way in examining Makinson's conditions in the context of ordered theory presentations over classical logic. In order to emulate variation of the facts with a fixed set of defaults, we can consider the consequences of the following ordered presentation with Δ fixed and Φ varying:

$$\begin{array}{c}(\Delta)\\ \uparrow \\ \Phi\end{array}$$

This is the OTP Δ with Φ appended at the bottom, which we will write as $\Delta * \Phi$ until the end of this section. We can think of this OTP as a way of representing that which in other default formalisms might be called 'the theory Φ with defaults Δ'. Notice that Δ is itself an OTP; that is, we are still allowing defaults with different priorities. Using this idea we can define a consequence relation $\mid\sim$ which embodies the defaults, as in Makinson's work. The obvious thing to do is to let $\Phi \mid\sim \psi$ mean $\Delta * \Phi \models \psi$. However, we know from proposition 2.20 that $\bot$ does not have its classical behaviour in the context of OTPs. We can get improved results by setting:

Definition 4.4 $\Phi \mid\sim \psi$ if $\bigwedge \Phi = \bot$ or $\Delta * \Phi \models \psi$.

That is to say, if Φ is contradictory then it entails everything; otherwise, it entails just what the illustrated OTP entails.

Recall that the technique of model ordering which originates in McCarthy's first circumscription paper [McC80] has been generalised in various ways [Sho88, Bes88, KLM90, Vel91, etc.]. It is further generalised by Makinson in [Mak92], where he proves that preferential model structures which satisfy a condition which he calls 'stopperedness' generate inference relations which satisfy each of the conditions on inference relations defined above. We therefore need simply show that the relation $\mid\sim$ of definition 4.4 *is* such a relation to prove that

Proposition 4.5 $\vdash$ satisfies supraclassicality, inclusion, cumulativity, and distributivity.

This is indeed the case, the condition of stopperedness following from our proposition 2.19. As before, the full proofs are spelled out in [Rya92a]. We thus have shown that OTPs over classical logic can yield a default inference relation in the sense of Makinson, with good formal properties.

5 Belief Revision

Ordered theory presentations have significant application in belief revision, whose basic question is: how should new information be incorporated into a belief state to result in a belief state which contains the new information and as much of the original belief state as is consistent? The best-known work on this subject is called the AGM theory (after its originators, C. Alchourrón, P. Gärdenfors and D. Makinson) [Gär88].

A full account of the AGM theory and the belief revision functions obtained from ordered theory presentations is given in [Rya92b] and [Rya92a]. We summarise our main findings below, but the interested reader should consult the more expansive references.

The AGM theory represents belief states as deductively-closed sets of sentences. Let K be such a belief state and ϕ a sentence. The revision of K by ϕ is written $K*\phi$. The AGM theory sets out eight postulates which a belief revision function $*$ must satisfy, known as K1–K8. (These may be found in any of the standard references; we repeat them in a generalised form below.)

We argue, however, that the eight axioms are neither *sound* not *complete* with respect to intuitively rational belief revision. Of course such a statement is necessarily imprecise, because 'intuitively rational' belief revision is not amenable to mathematical description. The argument to show lack of soundness is to give 'counterexamples' to K4 and K8, which are given later in the paper. My argument against completeness is the following proposition, which shows that K1–8 admit revision functions which fail to preserve any of the original belief state in many cases.

Proposition 5.1 The revision function

$$K * \phi = \begin{cases} \mathrm{Cn}\{K \cup \phi\} & \text{if } \neg\phi \notin K \\ \mathrm{Cn}\{\phi\} & \text{otherwise} \end{cases}$$

satisfies axioms K1–8.

In addition to this undesirable property of the AGM system, there is the further fact of that system that one cannot perform revision more than once. Repeated or iterated revision is not constrained by the axioms, and none of the models proposed for the AGM axioms (like revision by selection functions and epistemic entrenchment [Gär88]) define it[5].

Before considering how belief revision works in the context of OTPs, we have to generalise the AGM axioms. As things stand, they rely on a particular representation of belief states (namely, deductively closed sets of sentences). Therefore, direct comparison with theories of belief revision which use other representations of belief states is impossible. To overcome this we can we rewrite the axioms in a more general way, which assumes only the following:

1. A set of belief states, together with a subset of 'contradictory' belief states.
2. A function $*$ (revision) which takes a belief state and a sentence to a belief state;
3. A function $|\cdot|$ (extension) which takes a belief state and returns the set of sentences true in it.

Here are the axioms rewritten in this way. We will write $\mathcal{K}$ for a typical 'abstract' belief state.

K1 $\mathcal{K} * \phi$ is a belief state;
K2 $\phi \in |\mathcal{K} * \phi|$;
K3 $|\mathcal{K} * \phi| \subseteq |\mathcal{K}| + \phi$;
K4 If $\neg\phi \notin |\mathcal{K}|$ then $|\mathcal{K}| + \phi \subseteq |\mathcal{K} * \phi|$;
K5 $\mathcal{K} * \phi$ is contradictory implies $\phi = \bot$;
K6 If $\models \phi \leftrightarrow \psi$ then $|\mathcal{K} * \phi| = |\mathcal{K} * \psi|$;
K7 $|\mathcal{K} * (\phi \wedge \psi)| \subseteq |\mathcal{K} * \phi| + \psi$;
K8 If $\neg\psi \notin |\mathcal{K} * \phi|$ then $|\mathcal{K} * \phi| + \psi \subseteq |\mathcal{K} * (\phi \wedge \psi)|$.

We now turn to the belief revision theory offered by the OTP framework. We define

$$\text{belief states} = \text{ordered theory presentations} \cup \{\bot\}.$$

As belief revision gives rise only to *linear* OTPs we can write them with a more succinct notation. The OTP of example 2.2 will be written $[p, q, \neg p \vee \neg q]$.

Revision on these belief states is defined as follows:

$$\Gamma * \phi = \begin{cases} \bot & \text{if } \phi = \bot \\ [\phi] & \text{if } \phi \neq \bot \text{ and } \Gamma = \bot \\ \Gamma \text{ appended with } \phi & \text{otherwise} \end{cases}$$

The general case, therefore, is that we simply append the revising sentence. In other words, belief states are (usually) just revision histories.

[5] For the expert reader, we remark that there are proposals to allow repeated revision using EE orderings, either by keeping a single EE ordering for all belief states or assuming the existence of a function which, for every belief state, gives an EE ordering [Rot, Sch91]. But as neither the single ordering nor this function is itself revised in the course of belief revisions, it is easy to find examples which are in contradiction with intuitions about iterated belief change [Han91].

We argue that this function performs intuitively correct belief revision. As well as allowing repeated revision, it has the 'persistence' requirement mentioned above. However, this belief revision function, while satisfying $\mathcal{K}1$, $\mathcal{K}2$, $\mathcal{K}3$, $\mathcal{K}5$, $\mathcal{K}6$ and $\mathcal{K}7$, fails to satisfy $\mathcal{K}4$ and $\mathcal{K}8$. The counterexample to these two is given by example 2.3 and figure 3 of this paper, by setting $\mathcal{K} = [p \wedge q \wedge r, \neg p \vee \neg q \vee \neg r]$ and $\phi = (p \leftrightarrow q) \vee \neg r$ for K4; and $\mathcal{K} = [p \wedge q \wedge r]$, $\phi = \neg p \vee \neg q \vee \neg r$ and $\psi = (p \leftrightarrow q) \vee \neg r$ for K4. An explanation in both technical and intuitive terms of this counterexample may be found in the references already cited.

Acknowledgements

I am very grateful to Murray Shanahan for useful discussions, particularly concerning the relationships with Circumscription.

References

[AIJ80] Artificial Intelligence. Special Issue on Non-Monotonic Logic, volume 13, 1980.

[Bak91] A. B. Baker. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, 49:5–23, 1991.

[Bes88] P. Besnard. The preferential-models approach to non-monotonic logics. In P. Smets, A. Mamdani, D. Dubois, and H. Prade, editors, *Non-standard Logics for Automated Reasoning*. Academic Press, 1988.

[Bib85] W. Bibel. Methods of automated reasoning. In J. Bibel, editor, *Fundamentals in Artificial Intelligence*. Lecture Notes in Computer Science 232, Springer Verlag, 1985.

[Bre89] G. Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *Proc. International Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1043–1048. Morgan Kaufmann, 1989.

[CG88] M. R. B. Clarke and D. M. Gabbay. An intuitionistic basis for non-monotonic logic. In P. Smets, A. Mamdani, D. Dubois, and H. Prade, editors, *Non-standard Logics for Automated Reasoning*. Academic Press, 1988.

[CK90] C. C. Chang and H. K. Keisler. *Model Theory*. North-Holland, third edition, 1990.

[Gab91] D. M. Gabbay. Theoretical foundations for non-monotonic reasoning. part 2: Structured non-monotonic theories. In *Proc. Third Scandanavian Conference on Artificial Intelligence (SCAI'91)*, 1991.

[Gär88] P. Gärdenfors. *Knowledge in Flux: Modelling the Dynamics of Epistemic States*. MIT Press, 1988.

[Han91] S. O. Hansson. *Belief Base Dynamics*. PhD thesis, Department of Philosophy, Uppsala University, 1991.

[HM86] S. Hanks and D. McDermott. Default reasoning, non-monotonic logics and the frame problem. In *Proc. Fifth National Conference on Artificial Intelligence (AAAI)*, pages 328–333, 1986.

[Kau86] H. Kautz. The logic of persistence. In *Proc. Fifth National Conference on Artificial Intelligence*, pages 401–405, 1986.

[KLM90] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.

[Leh89] D. Lehmann. What does a conditional knowledge base entail? In *Proc. First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*. Morgan Kaufmann, 1989.

[Lif85] V. Lifschitz. Computing circumscription. In *Ninth International Joint Conference on Artificial Intelligence*, pages 121–127, 1985.

[Lif87] V. Lifschitz. Pointwise circumscription. In M. L. Ginsberg, editor, *Readings in Non-monotonic Logic*. Morgan Kaufmann, 1987.

[Mak88] D. Makinson. General theory of cumulative inference. In M. Reinfrank, J. de Kleer, and M. L. Ginsberg, editors, *Non-monotonic Reasoning*. Lecture Notes in Artificial Intelligence 346, Springer-Verlag, 1988.

[Mak92] D. Makinson. General patterns in non-monotonic reasoning. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence*. Oxford University Press, 1992.

[McC80] J. McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[Poo88] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.

[Rot] H. Rott. Preferential belief change using generalised epistemic entrenchment. Konstanzer Berichte zur Logik und Wissenschaftstheorie 15.

[Rya91] M. D. Ryan. Defaults and revision in structured theories. In *Proc. Sixth IEEE Symposium on Logic in Computer Science (LICS)*, pages 362–373, 1991.

[Rya92a] M. Ryan. *Ordered Presentations of Theories: Default Reasoning and Belief Revision*. PhD thesis, Department of Computing, Imperial College, 1992. Copies available from author.

[Rya92b] M. D. Ryan. Belief revision and ordered theory presentations. In P. Dekker and M. Stokhof, editors, *Proc. Eighth Amsterdam Colloquium on Logic*, 1992.

[Sch91] K. Schlechta. Some results on theory revision. In A. Fuhrmann and M. Morreau, editors, *The Logic of Theory Change*. Lecture Notes in Artificial Intelligence 465, Springer Verlag, 1991.

[Sha92] M. Shanahan. A circumscriptive calculus of events. Technical report, Imperial College Department of Computing, 1992.

[Sho88] Y. Shoham. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press, 1988.

[Vel91] F. Veltman. Defaults in update semantics. Technical Report LP-91-02, Institute for Language, Logic and Information, Amsterdam, 1991.

Rank-based systems: A simple approach to belief revision, belief update, and reasoning about evidence and actions.

Moisés Goldszmidt < *moises@cs.ucla.edu* >
Judea Pearl < *judea@cs.ucla.edu* >
Cognitive Systems Laboratory, Computer Science Department,
University of California, Los Angeles, CA 90024

Abstract

We describe a ranked-model semantics for if-then rules admitting exceptions, which provides a coherent framework for many facets of evidential and causal reasoning. Rule priorities are automatically extracted form the knowledge base to facilitate the construction and retraction of plausible beliefs. To represent causation, the formalism incorporates the principle of *Markov shielding* which imposes a stratified set of independence constraints on rankings of interpretations. We show how this formalism resolves some classical problems associated with specificity, prediction and abduction, and how it offers a natural way of unifying belief revision, belief update, and reasoning about actions.

1 Introduction

This paper is a culmination of several attempts to give conditional knowledge bases (with exceptions) empirical semantics in terms of infinitesimal probabilities, to be regarded as qualitative abstractions of an agent's experience. This semantics can be described in terms of rankings on models, where higher ranked models stand for more surprising (or less likely) situations. At the heart of this formulation is the concept of *default priorities*, namely, a natural ordering of the conditional sentences which can be derived automatically from the knowledge base and which can be used to answer queries without computing explicit rankings of worlds or formulas. The result is a model-theoretic account of plausible beliefs that, as in classical logic, are qualitative and deductively closed and, as in probability, are subject to retraction and to varying degrees of firmness.

The first part of this paper (Section 2) gives a brief summary of this rank-based semantics and describes a query-answering system called system-Z^+ which embodies this semantics in effective computational procedures. The main thrust of the paper (Section 3) is the introduction, within the basic framework of ranking systems, of a simple mechanism called *stratification* for the representation of causal relationships, actions, and changes.

The lack of a mechanism for distinguishing causal relationships from other kinds of associations has been a serious deficiency in most nonmonotonic systems [28], the classical illustration of which is given by the now-famous Yale Shooting Problem (YSP) [20]. In its simplified version, the YSP builds the expectation that if a gun is loaded at time t_0 and Fred is shot with the gun at time t_1, Fred should be dead at time t_2, despite the normal tendency of *being alive* to persist. Many formulations — including circumscription [26], default logic [34], rational closure [23], and conditional entailment [13] — reveal an alternative, perfectly symmetrical version of reality, whereby somehow the gun got unloaded and Fred is alive at time t_2.

The inclination to choose the scenario in which Fred dies is grounded in notions of directionality and asymmetry that are particular to causal relationships. In this paper we show that these notions can be derived from one fundamental principle, *Markov shielding*, which can be embodied naturally in preferential model semantics using the device of stratified rankings. Informally, the principle can be stated as follows:

- Knowing the set of causes for a given effect renders the effect independent of all prior events.

In the YSP, given the state of the gun at time t_1, the effect of the shooting can be predicted with total disregard for the gun's previous history.

We propose a probabilistically motivated, ranked-model semantics for rules of the form "typically, if cause_1 and ... and cause_n, then effect", which incorporates the above principle under the assumption that "causes" precede their "effects". As a by-product, our semantics exhibits another feature characteristic of causal organizations: *modularity*. Informally,

- Adding rules that predict future events cannot invalidate beliefs concerning previous events.

This is analogous to a phenomena we normally associate with causal mechanisms such as logical gates in electrical circuits, where connecting the inputs of a new gate to an existing circuit does not alter the circuit's behavior [7].

Although several remedies were proposed for the YSP within conventional nonmonotonic formalisms [35, 13, 37, 2, 24], the formalism we explore in this paper seeks to uncover remedies systematically from basic probabilistic principles [29, pp. 509–516]. We show that incorporating such principles in the qualitative context of world ranking yields useful results on several frontiers. In prediction tasks (such as the YSP), our formalism prunes the undesirable scenarios, without the strong commitment displayed by *chronological minimization* [35] and without the addition of *external* causal operators to the conditional interpretation of the rules [13]. In abduction tasks (such as when Fred is seen alive at t_2), our formalism yields plausible explanations for the facts observed (e.g., similar to [37], the gun must have been unloaded sometime before the shooting at t_1). This suggests that the principle of Markov shielding, by being grounded in probability theory (hence in empirical reality), can provide a coherent framework for the many facets of causation found in commonsense reasoning. Moreover, given the connection formed among causation, defaults, and probability, we can now ask not merely how to reason with a given set of causal assertions but also whether those assertions are compatible with a given stream of observations.

In the last part of this paper (Section 4) we demonstrate how rank-based systems can embody and unify the theories of belief revision [1] and belief updating [21], two theories of belief change that have been developed independently of research in default and causal reasoning. Basically, theories of belief change seek general principles for constraining the process by which a rational agent ought to incorporate a new piece of information ϕ into an existing set of beliefs ψ, regardless of how the two are represented and manipulated. Belief revision deals with new information obtained through new observations in a static world, while belief update deals with tracing changes in an evolving world, such as that subjected to the external influence of actions.

We show that system-Z^+ offers a natural embodiment of the principles of belief revision as formulated by Alchourrón, Gärdenfors and Makinson (AGM) [1], with the additional features of enabling the absorption of new conditional sentences and the verification of counterfactual sentences and nested conditionals. We then show that the addition of stratification to system-Z^+, by virtue of representing actions and causation, also provides the necessary machinery for embodying belief updates consistent with the principles proposed by Katsuno and Mendelzon (KM) [21].

2 Rankings and System-Z^+: Review

We assume throughout a finite set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of atomic propositions. The greek letters $\varphi, \psi, \sigma, \varphi$ will denote well-formed formulas (wff) built from the elements in $\mathcal{X}$. A *possible world* ω is a truth assignment to the propositions in $\mathcal{X}$. The satisfaction of a wff φ by an world ω is defined as usual and denoted by $\omega \models \varphi$. If ω satisfies φ then we say that ω is a model for φ.

A *defeasible conditional* or *default* is a formula "$\varphi \stackrel{\delta}{\rightarrow} \psi$", where φ and ψ are wffs (built from $\mathcal{X}$), "$\rightarrow$" is a new binary connective, and δ is a non-negative integer. The intended reading of $\varphi \stackrel{\delta}{\rightarrow} \psi$ is "typically, if φ then *expect* ψ (with strength δ)".[1] The connective "$\rightarrow$" imposes preferences among the possible worlds ω, requiring that if $\varphi \rightarrow \psi$, then ψ must be true in all the most preferred models for φ. In order to represent these preferences, we introduce ranking functions on the set Ω of possible worlds.

Definition 1 (Rankings) A **ranking function** κ is an assignment of non-negative integers to the elements in Ω, such that $\kappa(\omega) = 0$ for at least one $\omega \in \Omega$. We extend this definition to induce rankings on wffs:

$$\kappa(\varphi) = \begin{cases} \min_{\omega \models \varphi} \kappa(\omega) & \text{if } \varphi \text{ is satisfiable} \\ \infty & \text{otherwise} \end{cases} \quad (1)$$

Similarly, for a pair of wffs φ and ψ we define the conditional ranking $\kappa(\psi|\varphi)$ as

$$\kappa(\psi|\varphi) = \begin{cases} \kappa(\psi \wedge \varphi) - \kappa(\varphi) & \text{if } \kappa(\varphi) \neq \infty \\ \infty & \text{otherwise} \end{cases} \quad (2)$$

Preferences are associated with lower κ, and *surprise* or *abnormality* with higher κ. Thus, $\kappa(\psi) < \kappa(\varphi)$ if ψ is preferred to φ in κ, or equivalently, if φ is more abnormal (surprising) than ψ in κ. Intuitively, $\kappa(\psi|\varphi)$ stands for the degree of *surprise* or *abnormality* associated with finding ψ to be true, given that we already know φ. The inequality $\kappa(\neg\psi|\varphi) > \delta$ means that, given φ it would be surprising (i.e., abnormal) by at least $\delta + 1$ ranks to find $\neg\psi$, and it is equivalent to $\kappa(\psi \wedge \varphi) + \delta < \kappa(\neg\psi \wedge \varphi)$ which is precisely the constraint on worlds we attribute to $\varphi \stackrel{\delta}{\rightarrow} \psi$.

Definition 2 (Consistency) A ranking κ is said to be **admissible** relative to a given Δ, iff

$$\kappa(\varphi_i \wedge \psi_i) + \delta_i < \kappa(\varphi_i \wedge \neg\psi_i) \quad (3)$$

(equivalently $\kappa(\neg\psi_i|\varphi_i) > \delta_i$) for every rule $\varphi_i \stackrel{\delta_i}{\rightarrow} \psi_i \in \Delta$. A set Δ is **consistent** iff there exists an admissible ranking κ relative to Δ.

[1] The special case of $\delta = \infty$ corresponds to a *strict* conditional, to be denoted by $\Rightarrow$.

Consistency can be decided in $O(|\Delta|^2)$ satisfiability tests on the *material counterparts*[2] of the defaults in Δ, and it is independent of the δ-values assigned to the rules in Δ [15]. Eq. 3 echoes the usual interpretation of *defaults*, according to which ψ holds in all *minimal* models for φ. In our case, minimality is reflected in having the lowest rank. If we say that ω *falsifies* or *violates* a rule $\varphi \xrightarrow{\delta} \psi$ whenever $\omega \models \varphi \wedge \neg\psi$, the parameter δ can be interpreted as the minimal degree of surprise (or abnormality) associated with finding the rule $\varphi \xrightarrow{\delta} \psi$ violated, given that we know φ. In probabilistic terms, consistency guarantees that for every $\varepsilon > 0$, there exists a probability distribution P such that if $\varphi_i \xrightarrow{\delta_i} \psi_i \in \Delta$, then $P(\psi_i|\varphi_i) \geq 1 - c\varepsilon^{\delta_i}$ (see [17]).

2.1 The most normal ranking: κ^+

Given a set Δ, each admissible ranking κ induces a consequence relation $\mid\!\sim_\kappa$, where $\phi \mid\!\sim_\kappa \sigma$ iff $\kappa(\sigma \wedge \phi) < \kappa(\neg\sigma \wedge \phi)$. A straightforward way to declare σ as a plausible conclusion of Δ given ϕ would be to require $\phi \mid\!\sim_\kappa \sigma$ in all κ admissible with Δ. This leads to an entailment relation called ε-semantics [29], 0-entailment [31], and r-entailment [23], which is recognized as being too conservative. The approach we take here, following [31, 15, 23], is to select a distinguished admissible ranking, in our case κ^+, and declare σ as a plausible conclusion of Δ given ϕ, written $\phi \mid\!\sim_+ \gamma$, iff $\kappa^+(\phi \wedge \sigma) < \kappa^+(\phi \wedge \neg\sigma)$.[3] The distinguished ranking κ^+ assigns to each world the lowest possible rank permitted by the admissibility constraints of Eq. 3 (Def. 2), thus reflecting the assumption that, unless we are forced to do otherwise, each world is considered as normal (likely) as possible.

Definition 3 (The ranking κ^+) Let $\Delta = \{r_i \mid r_i = \varphi_i \xrightarrow{\delta_i} \psi_i\}$ be a consistent set of rules. κ^+ is defined as an admissible ranking function that is minimal in the following sense: Any other admissible ranking function must assign a higher ranking to at least one world and a lower ranking to none.

Theorem 4 ([17]) *Any consistent Δ has a unique minimal ranking κ^+ given by*

$$\kappa^+(\omega) = \begin{cases} 0 & \text{if } \omega \text{ does not falsify any rule in } \Delta, \\ \max_{\omega \models \varphi_i \wedge \neg\psi_i}[Z^+(r_i)] + 1 & \text{otherwise,} \end{cases} \quad (4)$$

*where $Z^+(r_i)$ is a set of integers defined on rules (*priorities*) which can be computed from Δ.*

Thus, the default rule priorities Z^+ constitute an economical way of encoding the ranking κ^+, linear in the size of Δ, from which the κ^+ of any world can be computed according to Eq. 4. In [17] we present an effective procedure, Procedure Z_rank, for computing Z^+, as well as answering queries. In the special case of a *flat* Δ, that is all δ's $= 0$, the procedure is as follows: We first identify all rules $r_i : \varphi_i \rightarrow \psi_i$ in Δ for which the formula

$$\varphi_i \wedge \psi_i \bigwedge_{j \neq i, r_j \in \Delta} \varphi_j \supset \psi_j \quad (5)$$

is satisfiable. Next we assign to these defaults priority $Z^+ = 0$, remove them from Δ, and repeat the process, assigning to the next set of defaults the priority $Z^+ = 1$, then $Z^+ = 2, \ldots$ and so on. Once Z^+ is known, the rank κ^+ of any wff ϕ is given by $\kappa^+(\phi) =$ minimum i such that

$$\phi \bigwedge_{j: Z^+(r_j) \geq i} \varphi_j \supset \psi_j \quad (6)$$

is satisfiable

Theorem 5 ([17]) *Given a consistent Δ, the computation of the Z^+ priorities requires $O(|\Delta|^2 \times \log|\Delta|)$ satisfiability tests. Moreover, given the Z^+ priorities, determining the ranking κ^+ of a wff ψ and the strength δ with which an arbitrary query σ is confirmed, given the information ϕ, that is $\phi \mid\!\sim^{\delta}_{+} \sigma$, requires $O(\log|\Delta|)$ satisfiability tests.*

Another important result implied by Eqs. 5 and 6 gives a method of constructing a propositional theory $Th(\phi)$ that implies all the conclusions γ that plausibly follow from a given evidence ϕ, i.e., $\phi \mid\!\sim_+ \gamma$. Such a theory is given by the formula

$$Th(\phi) = \bigwedge_{i: Z^+(r_i) \geq \kappa^+(\phi)} \varphi_i \supset \psi_i \quad (7)$$

Clearly, if the rules in Δ are of Horn form, computing the priority ranking Z^+, κ^+ of a given ψ, and deciding the plausibility δ of queries $(\phi \mid\!\sim^{\delta}_{+} \sigma)$ can be done in polynomial time [9]. The resulting system for default reasoning based on κ^+ and Z^+ is called system-Z^+ [15, 17].

System-Z^+ can also be used to reason with *soft* evidence or imprecise observations such as when the context ϕ of a query is not given with absolute certainty, and all we have is a testimony saying that "ϕ is supported to a degree n." In [17] we establish two strategies processing such reports. The first strategy, named J-conditionalization, is based on *Jeffrey's Rule of Conditioning* [30]. It interprets the report as specifying that "all things considered," the new degree of disbelief for $\neg\phi$ should be $\kappa'(\neg\phi) = n$. The second strategy, named L-conditionalization, is based on the *virtual evidence* proposal described in [29]. It interprets the report as specifying the desired *shift* in the degree of belief in ϕ, as warranted by that report alone and

[2] The material counterpart of $\varphi \xrightarrow{\delta} \psi$ is the wff $\varphi \supset \psi$.

[3] If we are concerned with the strength δ with which the conclusion is endorsed, then $\phi \mid\!\sim^{\delta}_{+} \gamma$ iff $\kappa^+(\phi \wedge \sigma) + \delta < \kappa^+(\phi \wedge \neg\sigma)$.

"nothing else considered". Both interpretations yield semi-tractable procedures (i.e., polynomial for Horn theories) for assessing the plausibility of σ, free from the computational difficulties that plague most non-monotonic systems.

Section 4.1 demonstrates how the computational procedures of system-Z^+ can be employed in the context of belief revision. Next we strengthen the admissibility condition with an additional requirement which gives a causal character to the defaults in Δ.

3 Stratified Rankings

Let $c_1, \ldots, c_m$ and e be literals over the elements of $\mathcal{X}$. A *rule* is defined as the default $c_1 \wedge \ldots \wedge c_m \rightarrow e$,[4] where the conjunction "$c_1 \wedge \ldots \wedge c_m$" is called the antecedent of the rule and "e" its consequent.[5]

Given $\mathcal{X}$ and a set Δ of rules, the *underlying characteristic graph* for $\langle \mathcal{X}, \Delta \rangle$, is the directed graph $\Gamma_{\langle \mathcal{X}, \Delta \rangle}$ such that there is a node v_i for each $x_i \in \mathcal{X}$, and there is a directed edge from v_i to v_j iff there is a rule R in Δ where x_i (or $\neg x_i$) is part of the antecedent of R, and x_j (or $\neg x_j$) is the consequent of R. We say that Δ is a *causal network* (or *network* for short) if $\Gamma_{\langle \mathcal{X}, \Delta \rangle}$ is acyclic (i.e., $\Gamma_{\langle \mathcal{X}, \Delta \rangle}$ is a DAG). If $v_r, \ldots, v_s$ are the parents of v_t in $\Gamma_{\langle \mathcal{X}, \Delta \rangle}$, then the set $\{x_r, \ldots, x_s\}$ is called the *parent set* of x_t and the set $\{x_r, \ldots, x_s\} \cup \{x_t\}$ is called a *family*. Intuitively, the parent set of an event e represents all the known causes for e. A network Δ induces a strict partial order "$\prec$" on the elements of $\mathcal{X}$ where $x_i \prec x_j$ iff there is a directed path from v_i to v_j in $\Gamma_{\langle \mathcal{X}, \Delta \rangle}$. We will use $\mathcal{O}(\mathcal{X})$ to denote any total order on the elements of $\mathcal{X}$ satisfying $\prec$.[6] Intuitively, $\prec$ represents a natural order on events where causes precede their effects.

Definition 6 (Stratified Rankings.) Given a network Δ, an admissible ranking κ, and an ordering $\mathcal{O}(\mathcal{X})$; let X_i $(1 \leq i \leq n)$ denote a literal variable taking values from $\{x_i, \neg x_i\}$, and let Par_{X_i} denote the conjunction $X_r \wedge \ldots \wedge X_s$ where $\{X_r, \ldots, X_s\}$ is the parent set of x_i. We say that κ is **stratified** for Δ under $\mathcal{O}(\mathcal{X})$, if for $2 \leq i \leq n$, and for any instantiation of the variables $X_1, \ldots, X_i$, we have

$$\kappa(X_i | X_{i-1} \wedge \ldots \wedge X_1) = \kappa(X_i | Par_{X_i}) \tag{8}$$

Eq. 8 says that in a stratified ranking the degree of (ab)normality of an event x_i given all its prior events must be equal to the degree of (ab)normality of x_i given just the set of events constituting its parent set. This condition of stratification is closely related to the Markovian independence conditions embodied in Bayes Networks (BN) [29]. A BN is a pair $\langle D, P \rangle$ where D is a DAG and P is a probability distribution. Each node v_i in D corresponds to a variable X_i in P, and P decomposes into the product:

$$P(X_n, \ldots, X_1) = \prod_{i=1}^{i=n} P(X_i | Par_{X_i}) \tag{9}$$

which, similarly to Eq. 8, incorporates the assumption that the parent set of any given variable X_i renders X_i probabilistically independent of all its predecessors (in the given ordering). Causal networks can in fact be regarded as an order of magnitude abstraction of BN's, where exact numerical probabilities are replaced by integer-valued levels of surprise (κ), addition is replaced by min, and multiplication is replaced by addition (see [17, 36, 32]). Note that Eq. 8 can be rewritten to mirror Eq. 9 as:[7]

$$\kappa(X_n, \ldots, X_1) = \sum_{i=1}^{i=n} \kappa(X_n | Par_{X_n}) \tag{10}$$

We shall show that this requirement augments admissible rankings with the properties of Markov shielding and modularity (see Theorems 8 and 9 below), that we normally attribute to causal organizations.

The following theorem states that the stratification criteria (Eq. 8) does not depend on the specific ordering $\mathcal{O}(\mathcal{X})$. This implies that in order to test whether a given ranking κ is stratified relative to a network Δ, it is enough to test Eq. 8 against *any* ordering $\mathcal{O}(\mathcal{X})$.

Theorem 7 *Given a network Δ, let $\mathcal{O}_1(\mathcal{X})$ and $\mathcal{O}_2(\mathcal{X})$ be two orderings of the elements in $\mathcal{X}$ according to Δ. If κ is stratified for Δ under $\mathcal{O}_1(\mathcal{X})$, then κ is stratified for Δ under $\mathcal{O}_2(\mathcal{X})$.*

3.1 c-entailment

Similar to the case of defaults (Sec 2.1), given a network Δ each stratified ranking κ defines a consequence relation $\mid\!\sim_\kappa$ where $\phi \mid\!\sim_\kappa \sigma$ iff $\kappa(\sigma \wedge \phi) < \kappa(\neg\sigma \wedge \phi)$ or if $\kappa(\phi) = \infty$. A consequence relation is said to be

[4] We only consider flat causal rules in this paper.

[5] The form $c_1 \wedge \ldots \wedge c_m \rightarrow e$ does not restrict the development of this paper but it clarifies the exposition. A causal rule may take on the general form $\alpha(c_1, \ldots, c_m) \rightarrow \beta(e_1, \ldots, e_n)$ where α and β are any Boolean formulae. Any $\alpha(c_1, \ldots, c_m)$ can be simulated by a set of simpler rules, each containing a conjunction of atomic antecedents. Moreover, any rule $\alpha(c_1, \ldots, c_m) \rightarrow \beta(e_1, \ldots, e_n)$ can be represented by the following set of rules: $\alpha(c_1, \ldots, c_m) \rightarrow e'$, $\beta(e_1, \ldots, e_n) \Rightarrow e'$, and $\neg\beta(e_1, \ldots, e_n) \Rightarrow \neg e'$, where e' is a dummy variable and $\Rightarrow$ is a *strict conditional*.

[6] Note that, in particular, any ordering $\mathcal{O}(\mathcal{X})$ induced by a topological sort on the nodes of $\Gamma_{\langle \mathcal{X}\Delta \rangle}$, where $x_i < x_j$ if v_i precedes v_j in the topological sort, satisfies $\prec$.

[7] An even coarser abstraction of Eq. 9 in the context of relational databases can be found in [7], where the stratification condition is imposed on relations and then used in finding backtrack free solutions for constraint satisfaction problems.

proper for $\phi \mid\!\sim_\kappa \sigma$ iff $\kappa(\phi) \neq \infty$. A network Δ **c-entails** σ given ϕ, written $\phi \mid\!\sim_\Delta \sigma$, iff $\phi \mid\!\sim_\kappa \sigma$ in every κ stratified for Δ, which is proper for $\phi \mid\!\sim_\kappa \sigma$. In other words, given Δ, we can expect σ from the evidence ϕ, iff the preference constraint conveyed by $\phi \rightarrow \sigma$ is satisfied by every stratified ranking for Δ. We remark that c-entailment is not to be interpreted as stating that ϕ is believed to *cause* σ. Rather, it expresses an *expectation* to find σ true in the context of ϕ, having given a causal character to the rules in Δ.

Since the set of stratified rankings is a subset of the admissible rankings, by the results in [22], all the inference rules that are sound for cumulative logics [25] and ε-semantics [13] are also sound for c-entailment [18]. These inference rules however, are known to be too weak to constitute a full account of plausible reasoning. The next two theorems provide additional inference power (reflecting the stratification condition) which emanates from the causal structure of Δ. They establish conditions under which these inference rules can be applied modularly to subsets $\Delta' \subset \Delta$ with the guarantee that the resulting inferences will hold in Δ.

Theorem 8 *Let Δ be a network, and let $\{p_r, \ldots, p_s\}$ be a set of literals corresponding to the parent set $\{x_r, \ldots, x_s\}$ of x_t (each p_i, $r \leq i \leq s$, is either x_i or $\neg x_i$). Let e_{x_t} denote a literal built on x_t, and let $\mathcal{Y} = \{y_1, \ldots, y_m\}$ be a set of atomic propositions such that no $y_i \in \mathcal{Y}$ is a descendant of x_t in $\Gamma_{(\mathcal{X},\Delta)}$. Let ϕ_Y be any wff built only with elements from $\mathcal{Y}$ such that $\phi_Y \wedge p_r \wedge \ldots \wedge p_s$ is satisfiable. If $p_r \wedge \ldots \wedge p_s \mid\!\sim_\Delta e_{x_t}$ then $\phi_Y \wedge p_r \wedge \ldots \wedge p_s \mid\!\sim_\Delta e_{x_t}$.*

Theorem 9 *Let $\Delta' \subset \Delta$ and $\mathcal{X}' \subset \mathcal{X}$ such that if $x' \in \mathcal{X}'$ then all the rules in Δ with either x' or $\neg x'$ as their consequent are also in Δ'. Let φ and ψ be two wffs built with elements from $\mathcal{X}'$. If $\varphi \mid\!\sim_{\Delta'} \psi$ then $\varphi \mid\!\sim_\Delta \psi$.*

These theorems confirm that stratified rankings exhibit the properties of Markov shielding and modularity. As a corollary to Theorem 9 it is easy to see that c-entailment is insensitive to *irrelevant* propositions, moreover, given two networks with no causal interaction, their respective sets of plausible conclusions will be independent of each other. To obtain a complete proof theory for c-entailment the four axioms of graphoids [29, Chapter 3] need to be invoked.[8] However, Theorems 8 and 9 cover the essence of these axioms and are sufficiently powerful for the purposes of this paper.

To demonstrate the behavior of the proposed formalism, and the usefulness of Theorems 8 and 9 as inference rules, consider the following example:[9]

[8]The conditional independence defined by $\kappa(X_3|X_2,X_1) = \kappa(X_3|X_2)$ is clearly a graphoid since κ represents infinitesimal probabilities.

[9]This example is isomorphic to the YSP [13].

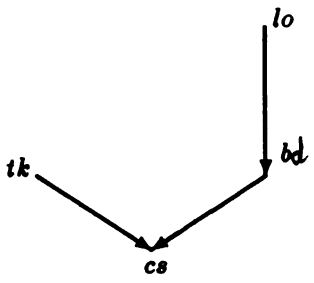


Figure 1: Underlying graph for the causal rules in Example 1

Example 1 (Dead battery) The network $\Delta = \{tk \rightarrow cs, tk \wedge bd \rightarrow \neg cs, lo \rightarrow bd\}$ encodes the information that "typically if I turn the ignition key the car starts", "typically if I turn the ignition key and the battery is dead the car will not start", and "typically if I leave the head lights on all night the battery is dead". The underlying graph for this network is depicted in Figure 1. Given Δ, and the fact the we left the head lights on all night, we don't expect the car engine to start once we turn the ignition key (i.e., $lo \wedge tk \mid\!\sim_\Delta \neg cs$). As in the case of YSP, an unintended scenario exists, in which the car engine actually starts and the battery is not dead after all. Table 1 contains an example of a stratified ranking for Δ, from which we can conclude that $lo \wedge tk \mid\!\sim_\Delta \neg cs$ as intended. A formal derivation of this conclusion is given in [18]. The key intermediate steps in this derivation rely on

κ	worlds
0	$(\neg lo, \neg bd, tk, cs)$, $(\neg lo, \neg bd, \neg tk, \neg cs)$
1	$(lo, bd, tk, \neg cs)$, $(lo, bd, \neg tk, \neg cs)$, $(\neg lo, bd, tk, \neg cs)$, $(\neg lo, bd, \neg tk, \neg cs)$
2	$(lo, \neg bd, tk, cs)$, $(lo, \neg bd, \neg tk, \neg cs)$
3	Rest of the ω's

Table 1: Stratified ranking for $\{tk \rightarrow cs, tk \wedge bd \rightarrow \neg cs, lo \rightarrow bd\}$.

Theorems 8 and 9:

- $tk \wedge lo \mid\!\sim_\Delta bd$. This follows from the proposition tk and applying Theorem 9 to the sub-network Δ' containing only the rule $lo \rightarrow bd$.
- $tk \wedge bd \mid\!\sim_\Delta \neg cs$ and $tk \wedge bd \wedge lo \mid\!\sim_\Delta \neg cs$. The former follows directly from ε-semantics, and the latter from applying Theorem 8 to the rule $tk \wedge bd \rightarrow \neg c$, and the proposition lo.

The next example presents a simple abduction (or backward projection) problem. We contrast the behavior of c-entailment with that of chronological minimization [35].

Example 2 (Unloading the gun.) Consider $\Delta = \{l_0 \rightarrow l_1, l_1 \rightarrow l_2, \ldots, l_{n-1} \rightarrow l_n\}$ standing for the various instances of "typically, if a gun is loaded at time t_i, then it is expected to remain loaded at time t_{i+1}" $(0 \leq i < n)$. We say that a rule $l_i \rightarrow l_{i+1}$ is *falsified* by ω iff $\omega \models l_i \wedge \neg l_{i+1}$; a stratified ranking κ relative to Δ can be constructed as follows:

$$\kappa(\omega) = \text{number of rules in } \Delta \text{ falsified by } \omega \qquad (11)$$

Given that the gun is loaded at t_0 and that it is found unloaded at time t_n (i.e., $l_0 \wedge \neg l_n$ is true), the scheme of chronological minimization will favor the somewhat counterintuitive inference that the gun remained loaded until t_{n-1} (i.e., $l_1 \wedge \ldots \wedge l_{n-1}$ is true). c-entailment on the other hand, only yields the weaker conclusion that the gun must have been unloaded any time within t_1 and t_{n-1} (i.e., $\neg(l_1 \wedge \ldots \wedge l_n)$), but the exact instant where the "unloading" of the gun occurs remains uncertain.

A full account of explanation and abduction using stratified rankings can be found in [18].

c-entailment and chronological minimization are expected to yield the same conclusions in problems of pure prediction, since enforcing ignorance of future events is paramount to the principle of modularity, which is inherent to c-entailment. They differ however in tasks of abduction, as demonstrated in Example 2. In this respect, c-entailment is closer to both *motivated action theory* [37] and *causal entailment* [13]. However, contrary to the motivated action theory, c-entailment automatically enforces specificity-based preferences, which are natural consequences of the conditional interpretation of rules.[10]

We end this section by discussing the *strict* version of a causal rule denoted by $\Rightarrow$, which will be useful in representing non-defeasible causal influences in Section 4. Semantically, strict rules impose the following constraints on the admissibility condition (Def. 2): for each $\varphi \Rightarrow \psi$ in the knowledge base,

$$\kappa(\psi \wedge \varphi) < \kappa(\neg\psi \wedge \varphi) = \infty, \quad \text{and } \kappa(\varphi) < \infty. \qquad (12)$$

Intuitively, a strict conditional voids interpretations that render its antecedent true and its consequent false by assigning them the lowest possible preference; a rank κ equal to infinity. The following are two properties of strict rules:

Theorem 10 *Let* $C_1 \wedge \ldots \wedge C_n \Rightarrow E \in \Delta$

1. **(Contraposition)** *If there exists a stratified ranking for* Δ *where* $\kappa(\neg E) < \infty$ *then* $\neg E \mid\!\sim_\Delta \neg(C_1 \wedge \ldots \wedge C_n)$
2. **(Transitivity)** *If* $\varphi \mid\!\sim_\Delta \psi$ *and* $\psi \models (C_1 \wedge \ldots \wedge C_n)$ *then* $\varphi \mid\!\sim_\Delta E$

These properties mirror the behavior of the material implication "$\supset$", however, they are not entirely identical to those governing a wff of propositional logic. In order for *contraposition* to hold, there is the additional consistency requirement that the negation of the consequent of the rule must be "possible" in at least one ranking. The precondition for *transitivity* to hold is governed by $\varphi \mid\!\sim_\Delta \psi$ and not by $\varphi \models \psi$. The semantic difference though between a strict rule $c \Rightarrow e$ and the wff $c \supset e$ is that the former expresses necessary hence permanent constraints while the latter expresses information bound to the current situation. Thus, the former participates in constraining the admissible rankings while the latter is treated as an "observation" formula $\neg c \vee e$, and can affect conclusions only by entering the antecedents of queries.[11]

3.2 c-consistency

Parallel to the notion of admissibility (Def. 2), we can define a notion of *c-consistency* as follows:

Definition 11 A network Δ is **c-consistent** iff there exists at least one stratified ranking κ for Δ

An example of a c-inconsistent network is the following: $\Delta = \{tk \rightarrow cs, tk \wedge bd \rightarrow \neg cs, tk \rightarrow x, x \rightarrow bd\}$.[12] The lack of an appropriate causal representation for this set of rules is not surprising. If we accept that tk causes cs, we should expect $\neg bd$ to hold by default when tk is true. On the other hand if there is a *causal path* from tk to bd, we should expect bd to hold in the context of tk. Note that this trouble case is admissible as shown by the ranking in Table 2.[13] This ranking depicts a situation in which the act of predicting the consequences of turning the key seems to protect the battery against the damage inflicted by x and such a flow of events is indeed contrary to the common understanding of causation.

Another admissible yet c-inconsistent set is $\Delta = \{a \Rightarrow c, b \Rightarrow \neg c\}$ which might possibly arise when we physically connect the outputs of two logic gates with conflicting functions. A stratified ranking for Δ would imply that

$$\kappa(a \wedge b) = \kappa(a) + \kappa(b) \qquad (13)$$

In other words the abnormality of a should be independent of the abnormality of b. However, if each time we

[10] We remark that the formalism in [37] deals with a much richer time ontology than the formalism presented here, and with a first-order language.

[11] See [14] for further discussion of strict rules vs. material implication.

[12] This is the network used in Example 1 augmented with two rules $tk \rightarrow x$ and $x \rightarrow bd$.

[13] This ranking is not stratified for Δ since $\kappa(bd \wedge x \wedge tk) = 2$, but $\kappa(bd|x) + \kappa(x|tk) + \kappa(tk) = 1$ which contradicts Eqs. 8 and 10.

κ	worlds
0	$(\neg tk, x, bd, \neg cs)$
1	$(tk, x, \neg bd, cs)$
2	$(tk, x, bd, \neg cs)$
3	Rest of the ω's

Table 2: Admissible ranking for $\{tk \rightarrow cs, tk \wedge bd \rightarrow \neg cs, tk \rightarrow x, x \rightarrow bd\}$.

observe a we should expect c, but each time we observe b we should expect not c, a and b must be mutually exclusive, hence negatively correlated events. Indeed, since $\kappa(a \wedge b \wedge c) = \kappa(a \wedge b \wedge \neg c) = \infty$ then $\kappa(a \wedge b) = \infty$ and Eq. 13 cannot be satisfied unless either a or b is permanently false, thus defying the "possible antecedent" requirement for strict rules (Eq. 12).

3.3 The most normal stratified ranking

In Section 2.1 we showed that the constraints provided by the default rules need to be supplemented with an additional assumption, so as to obtain a unique preferred ranking. The incorporation of this "most normal" assumption in the context of stratified rankings, results in a substantial increase of expressiveness as discussed in [18].

Note however, that contrary to the case of system-Z^+ (see Theorem 4), the most-normal stratified ranking may not be unique; for example the network $\Delta = \{a \rightarrow c, b \rightarrow \neg c\}$ has two minimal rankings [18]. Thus, we now need to define entailment in minimal rankings, denoted by $|\!\sim^*$, with respect to the consequence relations of all most-normal stratified rankings.

4 Belief Revision and Updating

In this section we demonstrate how the semantics of model ranking, together with the syntactic machinery developed for processing queries, can be applied to manage the tasks of belief revision and belief update. In both tasks we seek to incorporate a new piece of information ϕ into an existing set of beliefs ψ. In belief revision ϕ is assumed to be a piece of evidence while in update ϕ is treated as a change occurring by external intervention. We first apply the evidence handling capability of system-Z^+ to belief revision and then use stratified ranking and its representation of actions and causal relations to govern the dynamics of belief update.

4.1 Belief revision

AGM have advanced a set of postulates that have become a standard against which proposals for belief revision are tested [1]. The AGM postulates model epistemic states as deductively closed sets of (believed) sentences and characterize how a rational agent should change its epistemic states when new beliefs are added, subtracted, or changed. The central result is that the postulates are equivalent to the existence of a complete preordering of all propositions according to their degree of *epistemic entrenchment* such that belief revisions always retain more entrenched propositions in preference to less entrenched ones. Although the AGM postulates do not provide a calculus with which one can realize the revision process or even specify the content of an epistemic state [3, 10, 27], they nevertheless imply that a rational revision must behave as though propositions were ordered on some scale.

Spohn [36] has shown how belief revision conforming to the AGM postulates can be embodied in the context of ranking functions. Once we specify a single ranking function κ on possible worlds, we can associate the set of beliefs, with those propositions β for which $\kappa(\neg\beta) > 0$. It follows then that the models for the theory ψ representing our beliefs (written $Mods(\psi)$) consist of those worlds ω for which $\kappa(\omega) = 0$. To incorporate a new belief ϕ, one can raise the κ of all models of $\neg\phi$ relative to those of ϕ, until $k(\neg\phi)$ becomes (at least) 1, at which point the newly shifted ranking defines a new set of beliefs. This process of belief revision, which Spohn named α-conditioning (with $\alpha = 1$ for this particular case), represents the ranking equivalent of Jeffrey's rule of probability kinematics [29] and was shown to comply with the AGM postulates [12]. It follows then that the process of revising beliefs in all three forms of conditioning also obey the AGM postulates. Ordinary conditioning amounts to setting $\alpha = \infty$, J-conditioning amounts to $\alpha = J$, while L-conditioning calls for shifting the models of ϕ relative to those of $\neg\phi$ by L units of surprise. If we denote by $\kappa_\phi(\omega)$ the revised ranking after conditioning (with $\alpha = \infty$), then the dynamics of belief is governed by the following equation:

$$\kappa_\phi(\omega) = \begin{cases} \kappa(\omega) - \kappa(\phi) & \text{if } \omega \models \phi, \\ \infty & \text{otherwise.} \end{cases} \tag{14}$$

Accordingly, testing whether a given sentence β is believed after revision amounts to testing whether $\kappa_\phi(\neg\beta) > 0$ or, equivalently, whether $\kappa(\neg\beta|\phi) > 0$.

The unique feature of the system described in this paper is that the above test can be performed by purely syntactic terms, involving only the rules in Δ [17]. These computations are demonstrated in the following example.

Example 3 (Working students) The set $\Delta = \{s \rightarrow \neg w, s \rightarrow a, a \rightarrow w\}$ stands for "typically students don't work", "typically students are adults", and "typically adults work", respectively.[14] The Z^+ priorities on the rules (computed according to Eq. 5) are: $Z^+(a \rightarrow w) = 0$ and $Z^+(s \rightarrow \neg w) = Z^+(s \rightarrow a) = 1$,

[14] Note that all δ_i's are 0 for this example

from which the initial κ^+ ranking can be computed (Eq. 4), as depicted in Table 3. The rankings in Tables 4 and 5 show the revised rankings after observing an adult (κ_a) and a student (κ_s) respectively.

κ^+	Possible worlds
0	$(\neg s, a, w)$, $(\neg s, \neg a, w,)$, $(\neg s, \neg a, \neg w,)$
1	$(\neg s, a, \neg w)$, $(s, a, \neg w,)$
2	(s, a, w), $(s, \neg a, \neg w)$, $(s, \neg a, w)$

Table 3: Initial ranking for the student triangle in Example 3

κ_a^+	Possible worlds
0	$(\neg s, a, w)$
1	$(\neg s, a, \neg w)$, $(s, a, \neg w,)$
2	(s, a, w)

Table 4: Revised ranking after observing an adult

κ_s^+	Possible worlds
0	$(s, a, \neg w,)$
1	(s, a, w), $(s, \neg a, \neg w)$, $(s, \neg a, w)$

Table 5: Revised ranking after observing a student

The beliefs associated with these rankings can be computed from the worlds residing in $\kappa^+ = 0$. Thus, in κ_a^+ "an adult works", whereas in κ_s^+ "a student is an adult that does not work". These beliefs can be computed more conveniently by syntactic analysis of the rules and their Z^+ priorities, either by using Eq. 6, or by extracting from Δ a propositional theory that is maximally consistent with the observation using Eq. 7. For example, the beliefs associated with observing a student s are given by the theory $\{s, s \supset a, s \supset \neg w\}$. These two implications mirror the rules $s \rightarrow \neg w$ and $s \rightarrow a$ which are the unique set of rules that are maximally consistent with s.

4.2 Discussion and related work

There are several computational and epistemological advantages to basing the revision process on a finite set of conditional rules, and not on the beliefs, or on the rankings or the expectations that emanate from those rules. The number of propositions in one's belief set is astronomical, as is the number of worlds, while the number of rules is usually manageable.

This computational necessity has been recognized by several researchers. Nebel [27] adapted the AGM theory so that finite sets of *base* propositions mediate revisions. The basic idea in these syntax-based systems is to define a (total) priority order on the set of base propositions and to select revisions to be maximally consistent relative to that order as exemplified in the nonmonotonic systems of Brewka [5] and Poole [33] and in Example 3. Nebel has shown that such a strategy, can satisfy almost all the AGM axioms. Boutilier [3] has further shown that, indeed, the priority function Z^+ corresponds naturally to the epistemic entrenchment ordering of the AGM theory.[15]

Unfortunately, even Nebel's theory does not completely succeed at formalizing the practice of belief revision, as it does not specify how the priority order on the base propositions is to be determined. Although one can imagine, in principle, that the knowledge author specify this priority order in advance, such specification would be impractical, since the order might (and, as we have seen, should) change whenever new rules are added to the knowledge base. By contrast, system-Z^+ extracts both beliefs and rankings of beliefs automatically from the content of Δ; no outside specification of belief orderings is required.

Finally, and perhaps most significantly, system-Z^+ is capable of responding not merely to empirical observations but also to linguistically transmitted information such as conditional sentences (i.e., if–then rules). For example, suppose someone tells us that leaving the radio on also tends to render the battery dead; we add this new rule to our knowledge base (verifying first that the addition is admissible), recompute Z^+, and are prepared to respond to new observations or hearsay. In Spohn's system, where revisions are limited to α-conditioning, one cannot properly revise beliefs in response to conditional statements. The AGM postulates, too, are inadequate for describing revision due to incorporation of new conditionals.[16]

The ability to adopt new conditionals (as rules) also provides a simple semantics for interpreting nested conditionals (e.g., "If you wear a helmet whenever you ride a motorcycle, then you won't get hurt badly if you fall"[17]). Nested conditionals cease to be a mystery once we permit explicit references to default rules. The sentence "If $(a \rightarrow b)$ then $(c \rightarrow d)$" is interpreted as

> "If I add the default $a \rightarrow b$ to Δ, then the conditional $c \rightarrow d$ will be satisfied by the consequence relation $\mid\!\sim_{\Delta'}$ of the resulting knowledge base $\Delta' = \Delta \cup \{a \rightarrow b\}$".

[15] The proof in [3] considers the priorities Z^+ resulting from a *flat* set of rules as in system-Z [31]. Boutilier [4] also shows that an entrenchment ordering obeying the AGM framework obtains from the Z priorities of the negations of the material counterpart of rules.

[16] Gärdenfors [12] attempts to devise postulates for conditional sentences, but finds them incompatible with the Ramsey test (page 156-160).

[17] An example due to Calabrese (personal communication).

which is clearly a proposition that can be tested in the language of default-based ranking systems. Note the essential distinction between having a conditional sentence $a \rightarrow b$ explicitly in Δ versus having a conditional sentence $a \rightarrow b$ *satisfied* by the consequence relation of Δ. This distinction gets lost in systems that do not acknowledge defaults as the basis for ranking and beliefs.[18]

4.3 Belief update

The introduction of stratified ranking adds the capability for implementing a new type of belief changes, named *update* by Katsuno and Mendelzon, which result from external influences, and which act differently from those reflecting new evidence. Katsuno and Mendelzon [21] have shown that the AGM postulates are inadequate for describing changes caused by updates, for which they have proposed new sets of postulates. The basic difference between revision and update is that the latter permits changes in each possible world independently, as was proposed by Winslett [38].[19]

This type of belief change can be embodied in a stratified ranking system using the following device: For each instruction to "update the knowledge base by ϕ" we add a set of rules that simulates the action "$do(\phi)$, leaving everything else constant (whenever possible)", and then condition κ on the truth of $do(\phi)$. The following set of causal rules embody the intent of this action, where ϕ and ϕ' stand for "ϕ holds at t" and "ϕ holds at $t' > t$", respectively:[20]

$$\phi \rightarrow \phi' \tag{15}$$

$$\neg\phi \rightarrow \neg\phi' \tag{16}$$

$$do(\phi) \Rightarrow \phi'. \tag{17}$$

The following example (adapted from Winslett [38]) demonstrates how this device differentiates between update and revision.

Example 4 (XOR-gate) A XOR Boolean gate $c = XOR(a, b)$ is examined at two different times. At time t, we observe the output $c = true$ and conclude that one of the inputs a or b must be true, but not both. At a later time t' we learn that b' is true (primed letters denote propositions at time t'), and we wish to change our beliefs (in a and a') accordingly. Naturally, this change should depend on how the truth of

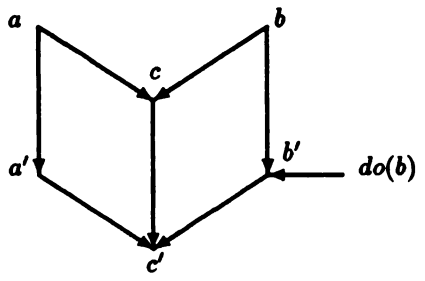


Figure 2: Graph depicting the causal dependencies in Example 4

b' is learned. If we learn b' by measuring the voltage on the b terminal of the gate, then we have a belief revision process on our hands, and we expect a' to be false. On the other hand, if we learn that b' is true as a result of physically connecting the b terminal to a voltage source, we no longer expect a' to be false, since we have no reason to believe that the output c has retained its truth value in the process.

In the stratified ranking formulation, the knowledge base corresponding to this example will consist of three components:

1. The functional description of the XOR gate at times t and t',

$$a \wedge b \Rightarrow \neg c \quad ; \quad \neg a \wedge \neg b \Rightarrow \neg c \tag{18}$$

$$a \wedge \neg b \Rightarrow c \quad ; \quad \neg a \wedge b \Rightarrow c, \tag{19}$$

   and an equivalent set of rules for a', b', c'.

2. The persistence rules: For every x in $\{a, b, c\}$,

$$x \rightarrow x' \quad ; \quad \neg x \rightarrow \neg x'. \tag{20}$$

3. The action $do(b)$, which represents the external influence on b':

$$do(b) \Rightarrow b'. \tag{21}$$

The underlying graph for the network Δ corresponding to this knowledge base is depicted in Figure 2.

Initially, after observing c, our evidence consists only of c. The minimal stratified ranking κ_c for a Δ consisting of rules in Eqs. 18-21 is depicted in Table 6. To represent belief revision, we add b' to our evidence set and query whether $c \wedge b' \mid\!\sim^*_\Delta \neg a'$.[21] In contrast, to represent belief update, we add $do(b)$ to our evidence set and query whether $(c \wedge b' \wedge do(b)) \mid\!\sim^*_\Delta \neg a'$.

It can be shown that the first query is answered in the affirmative, as the second in the negative. The left-hand side of Table 7 shows the ranking resulting from

[18] Belief revision systems proposed in the database literature [11, 6] suffer from the same shortcoming. In that context-defaults represent integrity constraints with exceptions.

[19] In the language of Bayesian networks, the difference between updates and revisions parallels the distinction between causal and evidential information [28].

[20] The two persistence rules, Eqs. 15 and 16, are presumed to apply between any two atomic propositions at two successive times.

[21] Recall that $\mid\!\sim^*_\Delta$ denotes the consequence relation of the minimal stratified ranking for Δ (see Sec. 3.3), which is unique for this example.

κ_c	$\neg do(b)$	$do(b)$
0	$(\neg a, b, \neg a', b'), (a, \neg b, a', \neg b')$	
1	$(\neg a, b, a', b'), (a, \neg b, \neg a', \neg b'), (a, \neg b, a', b')$	$(\neg a, b, \neg a', b'), (a, \neg b, a', b')$
2	$(\neg a, b, a', \neg b'), (a, \neg b, \neg a', b'),$	$(\neg a, b, a', b'), (a, \neg b, \neg a', b')$
∞	models for $\neg c$	models for $\neg c$

Table 6: Minimal stratified ranking for Example 4 after c is observed

κ_c	Revision $\kappa_c(\omega \mid b')$	Update $\kappa_c(\omega \mid do(b))$
0	$(\neg a, b, \neg a', b')$	$(\neg a, b, \neg a', b'), (a, \neg b, a', b')$
1	$(\neg a, b, a', b'), (a, \neg b, a', b')$	$(\neg a, b, a', b'), (a, \neg b, \neg a', b')$
2	$(a, \neg b, \neg a', b'),$	
∞	models for $\neg b'$	models for $\neg do(b)$

Table 7: Rankings after observing b, and after "doing" b

the revision of the ranking in Table 6 by b' (first query), while the right-hand side shows the ranking of worlds after updating by $do(b)$ (second query). Note that in the revised ranking the only world in the zero rank is a model for $\neg a'$, while the updated ranking shows an additional world which is a model for a' (the state of the output c in this world changed as a consequence of the action). The action $do(b)$ establishes the truth of b' but has no effect on what we believe about the second input a'. Since neither a nor $\neg a$ were believed at t, they remain unbelieved at t'.

4.4 The dynamics of belief update

The example above demonstrates that, given a ranking κ and a network Δ, it is possible to predict a system's behavior under external interventions. For example, if we wish to inquire whether event e will hold true after we force some variable A to become true, we simply add to Δ the rule $do(a) \Rightarrow a$,[22] recompute the resulting stratified ranking κ', and compute $\kappa'(e|do(a))$. It can be shown [16] that there is a simple relation between $\kappa(e|a)$ and $\kappa'(e|do(a))$, which is best represented as a transformation between two ranking functions, $\kappa(\omega)$ and $\kappa'(\omega)$, the latter being an abbreviation of $\kappa'(w|do(a))$. We simply replace the term $\kappa(a|Par_A)$ in the sum of Eq. 10 with the term $\kappa(a|do(a))$, representing the new influence $do(a)$ that now governs a:

$$\kappa'(\omega) = \begin{cases} \kappa(w) - \kappa(a|Par_A(w)) & \text{if } \omega \models a. \\ \infty & \text{if } \omega \models \neg a. \end{cases} \quad (22)$$

In other words, the κ of each world ω satisfying a is reduced by an amount equal to the degree of surprise of finding $A = true$, given the realization of Par_A in ω (the κ of each world falsifying a is of course ∞). Such independent movement from world to world is shown in Example 4, where $\kappa(\omega)$ is depicted on the left-hand side of Table 6 and $\kappa'(\omega)$ is depicted on the right hand side of Table 7. If A has no *parents* (direct causes), then κ' is obtained by shifting the κ of each $\omega \models a$ by a constant amount $\kappa(a)$, as in ordinary conditioning, and $\kappa'(\omega)$ would be equal to $\kappa(\omega|a)$, as expected. However, when the manipulated variable has direct causes Par_A, the amount of shift would vary from world to world, depending on how surprising it would be (in that world) to find a happening naturally (without external intervention). For instance, if A is governed by persistence rules, $a(t-1) \rightarrow a(t), \neg a(t-1) \rightarrow \neg a(t)$, then worlds in which $a(t-1)$ hold will shift less than those in which $a(t-1)$ is false, because $a(t)$ is expected to hold in the former and not in the latter. Note that the amount of shift subtracted from $\kappa(\omega)$ is equal precisely to the fraction of surprise $\kappa(a|Par_A(\omega))$ that $A = true$ contributes to $\kappa(\omega)$ and that now becomes *explained away* (hence excusable) by the action $do(a)$.

4.5 Relation to KM postulates

It can be shown [16] that when the update by a formula ϕ is given as a conjunction of literals (representing concurrent or sequential actions), then the movement of worlds toward $\kappa = 0$ will yield a set of updated beliefs consistent with the KM postulates.[23] More specifically, for every world $\omega \models \neg\phi$ that is currently in

[22] We use lowercase to denote the instantiation of variable A to a truth value.

[23] Updates involving disjunctions require special treatment. If they are to be interpreted as a license to effect any change satisfying the disjunction, then the final state of belief is the union, taken over all disjuncts, of worlds that drift to $\kappa = 0$. In this interpretation, the instruction "make sure the box is painted either blue or white" will leave the box color unknown, even knowing that the box was white initially (contrary to the postulate (U2) of KM). However, if the intention is to effect no change as long as the disjunctive condition is satisfied, then the knowledge base should be augmented with an observation-dependent strategy "$do(\phi)$ when ϕ is not satisfied", instead of using the pure action $do(\phi)$. Conditioning on such a strategy again yields a belief set consistent with the KM postulates. The first interpretation is useful for discrediting earlier observations, for example, "I am not sure the em-

$\kappa = 0$ there is at least one image world $\omega^* \models \phi$, having $\kappa(\omega^*) > 0$ that will end up at $\kappa'(\omega^*) = 0$ according to Eq. 22. In an image world ω^*, every term $\kappa(x_i|Par_{X_i}(\omega^*)) > 0$ represents a violation of expectation that would be totally excusable were it caused by an external intervention such as ϕ. Intuitively, the image world corresponds to a scenario in which all the unexpected events are attributed to the intervention of ϕ but otherwise the world follows its natural, unperturbed course as dictated by the prediction of the causal theory.

That updates resulting from Eq. 22 comply with the KM postulates can be seen by the following consideration.[24] KM have shown that their axioms are equivalent to the existence of a function mapping each possible interpretation world ω to a partial preorder $\leq_\omega$, such that for any interpretation ω', if $\omega \neq \omega'$ then $\omega <_\omega \omega'$. Then the set of models for the update of a formula ψ (representing our current beliefs) by a formula ϕ, written $\psi \diamond \phi$, is found by taking the union of the minimal models for ϕ, with respect to each one of the pre-orders defined by the models for ψ:

$$Mods(\psi \diamond \phi) = \bigcup_{\omega \in Mods(\psi)} \min(Mods(\phi), \leq_\omega). \quad (23)$$

It is not hard to show that the image ω^* as described above is indeed a minimal element in the order $\leq_\omega$, defined as follows:

Definition 12 (World orderings) Let $\mathcal{O} = x_1, x_2, \ldots, x_n$ be any order of the variables that is consistent with the dag $\Gamma_{(\chi,\Delta)}$. Given three worlds ω, ω_1, and ω_2, we say that $\omega_1 \leq_\omega \omega_2$ iff the following conditions hold:

1. ω disagrees with ω_2 on a literal that is earlier (in $\mathcal{O}$) than any literal on which ω disagrees with ω_1.
2. If a tie occurs, then $w_1 \leq_\omega \omega_2$ if $\kappa(\omega_1) \leq k(w_2)$.

4.6 Related work

The connection between belief update and theories of action was noted by Winslett [38] and has been elaborated more recently by del Val and Shoham [8] using the situation calculus.

Unlike del Val and Shoham [8], we would not claim that "the KM-postulates need not be postulated at all, but can instead be derived analytically". While the KM postulates can indeed be derived from our formulations of actions, persistence, and causation, the interesting power of these postulates is that they cover a wide variety of such formulations, from a simple theory such as ours to the intricate machinery of the situation calculus. Our analysis offers the KM postulates an intuitive, model-theoretic support that is well grounded in probability theory, where the distinction between observations and actions can be formulated naturally and tractably. It also offers a simple unification of revision and update, since both are embodied in a conditioning operator, the former by conditioning on *observations* and the latter by conditioning on *actions*.

Grahne et. al. [19] showed that revision could be expressed in terms of an update operator in a language of introspection (intuitively, *observing* a piece of evidence has the same effect as *causing* the observer to augment her beliefs by that very evidence). Our analysis shows that the converse is also true: belief updates can be expressed in terms of a *conditioning* operator, which is normally reserved for belief revision. The intuition is that *acting* to produce a certain effect yields the same beliefs as *observing* that action performed. This translation is facilitated by the special status that the added *action* $\Rightarrow$ *effect* rules enjoy in stratified ranking, where actions are always represented as root nodes, independent of all other events except their consequences. This ensures that the immediate effects of those actions are explained away and do not reflect back on other events in the past. It is this stratification that produces the desired distinction between observing an action produce an effect and observing the effect without the action.[25]

Acknowledgements

This work was supported in part by NSF grant #IRI-9200918, AFOSR grant #900136, and MICRO grant #91-124. We thank C. Boutilier, D. Lehmann, and two referees for comments on an earlier draft of this paper.

References

[1] C. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[2] A. B. Baker. Nomonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, 49:5–23, 1991.

[3] C. Boutilier. Conditional logics for default reasoning and belief revision. Ph.D. dissertation, University of Toronto, 1992.

[4] C. Boutilier. What is a default priority? In *Proceedings of CCAI-92*, Vancounver, 1992.

ployee's salary is $50K$; it could be anywhere between $40K$ and $60K$".

[24] A formal proof can be found in [16].

[25] Note that update cannot be expressed in terms of the AGM operators of revision and contraction, because it is impossible to simulate with these operators the acceptance of a new conditional $do(\phi) \Rightarrow \phi$, so that the acceptance of $do(\phi)$ is treated differently than the acceptance of ϕ. Similarly, update cannot be formulated in Spohn's system, because the identity of the image world ω^* cannot be described in terms of the initial ranking alone; it requires the causal theory Δ.

[5] G. Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *Proceedings of IJCAI-89*, Detroit, 1989.

[6] M. Dalal. Investigations into a theory of knowledge base revision: Preliminary report. In *Proceedings of AAAI-88*, pages 475–479, 1988.

[7] R. Dechter and J. Pearl. Directed constraint networks: A relational framework for causal modeling. In *Proceedings of IJCAI-91*, Australia, 1991.

[8] A. del Val and Y. Shoham. Deriving properties of belief update from theories of acion. In *Proceedings of AAAI-92*, pages 584–589, San Jose, California, 1992.

[9] W. Dowling and J. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 3:267–284, 1984.

[10] J. Doyle. Rational belief revision (preliminary report). In *Proceedings of Principles of Knowledge Representation and Reasoning*, pages 163–174, Cambridge, Massachusetts, 1991.

[11] R. Fagin, J. D. Ullman, and M. Vardi. On the semantics of updates in databases. In *Proceedings of the 2nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 352–365, 1983.

[12] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States.* MIT Press, Cambridge, 1988.

[13] H. A. Geffner. Default reasoning: Causal and conditional theories. MIT Press, Cambridge, 1991.

[14] M. Goldszmidt and J. Pearl. On the consistency of defeasible databases. *Artificial Intelligence*, 52:121–149, 1991.

[15] M. Goldszmidt and J. Pearl. System Z^+: A formalism for reasoning with variable strength defaults. In *Proceedings of AAAI-91*, pages 394–404, Anaheim, CA, 1991.

[16] M. Goldszmidt and J. Pearl. Dynamics of belief update. Technical Report TR-190, University of California Los Angeles, Cognitive Systems Lab., Los Angeles, 1992, (In preparation).

[17] M. Goldszmidt and J. Pearl. Reasoning with qualitative probabilities can be tractable. In *Proceedings of the 8th Conference on Uncertainty in AI*, Stanford, 1992.

[18] M. Goldszmidt and J. Pearl. Stratified rankings for causal relations. In *Proceedings of the Fourth International Workshop on Nonmonotonic Reasoning*, Vermont, 1992.

[19] G. Grahne, A. Mendelzon, and R. Reiter. On the semantics of belief revision systems. In *Proceedings of TARK-92*, pages 132–142, Monterrey, CA, 1992.

[20] S. Hanks and D. McDermott. Non-monotonic logics and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[21] H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 387–394, Boston, 1991.

[22] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.

[23] D. Lehmann. What does a conditional knowledge base entail? In *Proceedings of Principles of Knowledge Representation and Reasoning*, pages 212–222, Toronto, 1989.

[24] V. Lifschitz. Formal theories of action. In M. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*, pages 410–432. Morgan Kaufmann, San Mateo, 1987.

[25] D. Makinson. General theory of cummulative inference. In M.Reinfrank, J. de Kleer, M. Ginsberg, and E. Sandewall, editors, *Non-monotonic Reasoning*. Springer-Verlag, Lecture Notes on Artificial Intelligence 346, Berlin, 1989.

[26] J. McCarthy. Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[27] B. Nebel. Belief revision and default reasoning: Syntax-based approaches. In *Proceedings of Principles of Knowledge Representation and Reasoning*, pages 417–428, Cambridge, Massachusetts, 1991.

[28] J. Pearl. Embracing causality in formal reasoning. *Artificial Intelligence*, 35:259–271, 1988.

[29] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[30] J. Pearl. Jeffrey's rule, passage of experience and neo–Bayesianism. In H. Kyburg, R. Loui, and G. Carlson, editors, *Defeasible Reasoning and Knowledge Representation*, pages 121–135. Kluwer Publishers, San Mateo, 1990.

[31] J. Pearl. System Z: A natural ordering of defaults with tractable applications to default reasoning. In R. Parikh, editor, *Proceedings of TARK-90*, pages 121–135. Morgan Kaufmann, San Mateo, CA, 1990.

[32] J. Pearl. Epsilon-semantics. In *Encyclopedia of Artificial Intelligence*, pages 468–475. Wiley Interscience, New York, 1992. Second Edition.

[33] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.

[34] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[35] Y. Shoham. *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence.* MIT Press, Cambridge, Mass., 1988.

[36] W. Spohn. Ordinal conditional functions: A dynamic theory of epistemic states. In W. L. Harper and B. Skyrms, editors, *Causation in Decision, Belief Change, and Statistics*, pages 105–134. Reidel, Dordrecht, Netherlands, 1987.

[37] L. Stein and L. Morgenstern. Motivated action theory: A formal theory of causal reasoning. In *Proceedings of AAAI-88*, pages 518–523, 1988.

[38] M. Winslett. Reasoning about action using a possible worlds approach. In *Proceedings of AAAI-88*, pages 89–93, 1988.

Representing Default Rules in Possibilistic Logic

Salem BENFERHAT - Didier DUBOIS - Henri PRADE
Institut de Recherche en Informatique de Toulouse - C.N.R.S.
Université Paul Sabatier, 118 route de Narbonne
31062 TOULOUSE Cedex - France

Abstract

A key issue when reasoning with default rules is how to order them so as to derive plausible conclusions according to the more specific rules applicable to the situation under concern, to make sure that default rules are not systematically inhibited by more general rules, and to cope with the problem of irrelevance of facts with respect to exceptions. Pearl's system Z enables us to rank-order default rules. In this paper we show how to encode such a rank-ordered set of defaults in possibilistic logic. We can thus take advantage of the deductive machinery available in possibilistic logic. We point out that the notion of inconsistency tolerant inference in possibilistic logic corresponds to the bold inference $\vdash_1$ in system Z. We also show how to express defaults by means of qualitative possibility relations. Improvements to the ordering provided by system Z are also proposed.

1. INTRODUCTION

In the last five years many works dealing with the formal study of non-monotonic reasoning have appeared. These works go beyond early attempts which proposed specific non-monotonic reasoning systems because they now address the more fundamental issue of determining the natural properties of a non-monotonic consequence relation, likely to achieve a satisfactory treatment of plausible reasoning in the presence of incomplete or partially inconsistent information. Typical of this kind of research are the works of Gabbay (1985), Makinson (1989), Makinson and Gärdenfors (1991), Gärdenfors (1991) and Kraus et al. (1990). It is now widely acknowledged that the semantics of many non-monotonic logics can be described in terms of a preference relation on possible worlds (or interpretations) as first claimed by Shoham (1988). At the same time, logics of uncertainty such as probabilistic and possibilistic logics turned out to be related to non-monotonic reasoning. Pearl (1988) has suggested that Adams (1975)' logic of infinitesimal probabilities was a good basis for non-monotonic reasoning, and indeed the core axioms of non-monotonic reasoning are present in Adams' logic. In order to get a less conservative inference, Pearl (1990) has introduced system Z which computes a natural priority ordering among defaults and enables the problem of irrelevance (from $p \mathrel{|\!\sim} q$ deduce $p \wedge r \mathrel{|\!\sim} q$ for irrelevant r) to be coped with. This is again a well-behaved non-monotonic reasoning system.

Independently, two of the authors of this paper have developed, since 1987, possibilistic logic, a weighted logic that handles valued formulas of classical logic, where the weights express levels of certainty belonging to a totally ordered set (usually taken as the unit interval); see (Dubois and Prade, 1987). This logic is capable of coping with partial inconsistency, and the inference machinery turns out to express preferential entailment as in Shoham (Dubois and Prade, 1991b). Hence possibilistic logic is another system of non-monotonic logic that handles a total ordering of interpretations.

In this paper, we pursue the preliminary investigations of Dubois and Prade (1991a, b, c) by relating possibilistic logic and system Z of Pearl (1990). The following results are obtained, and explained:

- Given a set of defaults rank-ordered by system Z, these defaults can be encoded in possibilistic logic. The corresponding possibilistic knowledge base exactly yields the same deductions as the set of defaults ordered by system Z;
- At the semantic level, the minimal ranking function on models defined in system Z is shown to be equivalent to the so-called least specific possibility distribution on models that is the basis of the semantics of possibilistic logic;
- Defaults rules of Adams-Pearl conditional logic can be directly encoded under the form of a partially defined qualitative possibility relation on formulas (or equivalently a partially defined expectation ordering, in the terminology of Gärdenfors and Makinson, 1992). The converse is true, and the two kinds of inference $\vdash_0$ and $\vdash_1$ (respectively conservative and bold) in system Z can be expressed in terms of completions of partially defined orderings of formulas. The minimum

specificity ordering is described and leads to the bold inference of system Z.
- A new approach to ordering defaults in system Z is proposed where shortcomings pertaining to property inheritance are tackled.

2. POSSIBILITY THEORY

Let Ω be a set of possible worlds. If needed, Ω can be identified with the finite set of interpretations of a propositional logic language. A possibility distribution is a mapping π from Ω to a totally ordered set S which is also a complete lattice. For simplicity we assume S = [0,1]. π is said to be normal if $\exists\ \omega \in \Omega$, such that $\pi(\omega) = 1$. By convention π represents some background knowledge about where the real world is; $\pi(\omega) = 0$ means that ω is not possible, and $\pi(\omega) = 1$ that nothing prevents ω from being the real world. When $\pi(\omega) > \pi(\omega')$, ω is a preferred candidate to ω' for being the real world. π is thus a convenient encoding of a preference relation that can embody concepts such as normality, typicality, consistency with available knowledge, qualitative likelihood, etc.

A possibility distribution leads to evaluate, in two respects, subsets of possible worlds as to their containing the real world:

- a degree of possibility $\Pi(p) = \sup\{\pi(\omega) \mid \omega \models p\}$ which evaluates the extent to which p is consistent with the available knowledge expressed by π (Zadeh, 1978); thus we have:
$$\forall p\ \forall q \quad \Pi(p \vee q) = \max(\Pi(p), \Pi(q))$$
- a degree of necessity (or certainty) $N(p) = \inf\{1 - \pi(\omega) \mid \omega \models \neg p\}$ which evaluates the extent to which p is entailed by the available knowledge.

The duality between certainty and possibility is expressed by $N(p) = 1 - \Pi(\neg p)$. Possibility and certainty degrees can thus be used to induce ordering relations on the set of formulas of a language, from the knowledge of a possibility distribution on possible worlds, considering a formula as the set of possible worlds where it is true. The converse is true as well. A qualitative necessity (resp.: possibility) relation on a Boolean algebra is defined for any pair of propositions p and q as $p \geq_C q \Leftrightarrow N(p) \geq N(q)$ (resp. $p \geq_\Pi q \Leftrightarrow \Pi(p) \geq \Pi(q)$). Necessity relations are complete transitive relations such that $\top >_C \bot$, where $\top$ and $\bot$ represent tautology and contradiction respectively, and $>_C$ the strict part of the C-ordering $\geq_C$ (we will similarly distinguish in notation strict and non-strict relations for other orderings in the following)

$$\top \geq p \geq_C \bot, \forall p$$
$$\text{and } p \geq_C q \Rightarrow r \wedge p \geq_C r \wedge q, \forall r.$$

In the finite case, the only numerical counterparts to necessity relations are necessity measures, such that $N(p \wedge q) = \min(N(p), N(q))$ (Dubois, 1986). Qualitative possibility relations are dual to necessity relations in the sense that $p \geq_C q \Leftrightarrow \neg q \geq_\Pi \neg p$. They satisfy the characteristic axiom $p \geq_\Pi q \Rightarrow r \vee p \geq_\Pi r \vee q, \forall r$.

Qualitative necessity relations are very close to epistemic entrenchment relations of Gärdenfors (1988); see (Dubois and Prade, 1991a).

Another important issue in possibility theory is the principle of minimum specificity. A possibility distribution π is said to be more specific than another π' if and only if $\pi < \pi'$. In other words, π is more informative than π'. Given a set of constraints restricting a feasible subset of possibility distributions, the best representative is the least specific feasible possibility distribution, which assigns the highest degree of possibility to each world, since it is the least committed one.

Lastly, a purely ordinal notion of conditioning can be defined for possibility and necessity measures, by means of an equation similar to Bayesian conditioning (Dubois and Prade, 1990)

$$\Pi(p \wedge q) = \min(\Pi(q \mid p), \Pi(p)) \qquad (1)$$

when $\Pi(p) > 0$. $\Pi(q \mid p)$ is defined as the greatest solution to (1) in accordance with the minimum specificity principle. It leads to

$$\Pi(q \mid p) = 1 \text{ if } \Pi(p \wedge q) = \Pi(q)$$
$$= \Pi(p \wedge q) \text{ otherwise}$$

when $\Pi(p) > 0$. If $\Pi(p) = 0$, $\Pi(q \mid p) = 1$, $\forall q$. The conditional necessity measure is simply defined as $N(q \mid p) = 1 - \Pi(\neg q \mid p)$. Thus $N(q \mid p) = N(\neg p \vee q)$ provided that $N(q \mid p) > 0$. This notion of conditioning contrasts with Dempster rule of conditioning, whereby min is changed into product in (1) viewing a possibility measure as a special case of Shafer (1976)'s plausibility function.

3. POSSIBILISTIC ENTAILMENT

Given a possibility distribution π on Ω, a notion of preferential entailment $\models_\pi$ can be defined in the spirit of Shoham (1988)'s proposal:

$$p \models_\pi q \Leftrightarrow$$
all the worlds which maximize π, among those which satisfy p, satisfy q.

We restrict this definition to propositions p such that $\Pi(p) > 0$. It can then be established that (Dubois and Prade, 1991b)

$p \models_\pi q$ if and only if $\{\omega \models p \mid \pi(\omega) = \Pi(p) > 0\} \subseteq \{\omega \models q\}$
if and only if $N(q \mid p) > 0$
if and only if $N(\neg p \vee q) > N(\neg p \vee \neg q)$
if and only if $\Pi(p \wedge q) > \Pi(p \wedge \neg q)$.

In this section we assume for simplicity that $\Pi(p) > 0$, $\forall\, p \neq \bot$. It has been shown in Dubois and Prade (1991b, c) that the inference relation $\models_\pi$ is non-monotonic. It was shown to satisfy the three axioms suggested by Gabbay (1985):

Cut: if $p \models_\pi q$ and $p \wedge q \models_\pi r$ then $p \models_\pi r$
Cautious Monotony (CM):
if $p \models_\pi q$ and $p \models_\pi r$ then $p \wedge q \models_\pi r$
OR: if $p \models_\pi r$ and $q \models_\pi r$ then $p \vee q \models_\pi r$.

Moreover the following properties also hold:

Restricted Reflexivity (RR): if $p \neq \bot$ then $p \models_\pi p$
Nihil ex Absurdo (NA): $\neg(\bot \models_\pi p)$
Left Logical Equivalence (LLE):
if $p \leftrightarrow q = \top$, $p \models_\pi r$ then $q \models_\pi r$
Right Weakening (RW):
if $\neg p \vee q = \top$ then $r \models_\pi p$ implies $r \models_\pi q$
AND: if $p \models_\pi q$ and $p \models_\pi r$ then $p \models_\pi q \wedge r$
Rational Monotony (RM):
if $\neg(p \models_\pi \neg q)$ and $p \models_\pi r$ then $p \wedge q \models_\pi r$.

Most of these properties, studied by Kraus et al. (1990) belong to what Gärdenfors and Makinson (1992) call "extended set of postulates for non-monotonic inference". There are slight differences however; supraclassicality ($p \models q$ implies $p \models_\pi q$) is not taken for granted, but changed into restricted reflexivity, and the NA assumption is present; namely, we do not allow inferences from a contradictory statement ($p = \bot$) in the sense of preferential entailment. While $\bot$ entails anything, it should preferentially entail nothing. This is consistent with the fact that $\Pi(p \mid \bot) = 1$, $\forall p$ for conditional possibility. All these properties trivially hold for possibilistic entailment, except for rational monotony which comes down to proving

if $N(q \mid p) > 0$ and $N(\neg r \mid p) = 0$ then $N(q \mid p \wedge r) > 0$.

Indeed, the two premisses are equivalent to $\Pi(p) > \Pi(p \wedge \neg q)$ and $\Pi(p) = \Pi(p \wedge r)$; then $\Pi(p \wedge r) > \Pi(p \wedge \neg q) = \max(\Pi(p \wedge \neg q \wedge \neg r), \Pi(p \wedge \neg q \wedge r))$ therefore $\Pi(p \wedge r) > \Pi(p \wedge r \wedge \neg q)$ which is equivalent to $N(q \mid p \wedge r) > 0$. Moreover, note that from $N(q \mid p) > 0$ and $N(q \mid p \wedge r) > 0$ we have respectively $N(q \mid p) = N(\neg p \vee q)$ and $N(q \mid p \wedge r) = N(\neg p \vee \neg r \vee q)$ and from $\neg p \vee q \vdash \neg p \vee \neg r \vee q$ we have $N(\neg p \vee \neg r \vee q) > N(\neg p \vee q)$; then we conclude that $N(q \mid p \wedge r) > N(q \mid p)$ also holds.

As properties for non-monotonic inference, it is obvious that RM implies CM of which it is a strong form. Moreover Kraus et al. (1990) have proved that the rule

Conditionalization (C): if $p \wedge q \models_\pi r$ then $p \models_\pi q \rightarrow r$

follows from LLE, RW, AND, OR, CM, and reflexivity; hence it holds here if $p \wedge q \neq \bot$ as a consequence of LLE, RW, AND, OR, RM, RR, NA. Using C, AND and RW, the Cut rule is also a consequence of these axioms. Lastly another property can also be derived, namely

Consistency Preservation (CP): $\neg(p \models_\pi \bot)$

Proof: If $p = \bot$, CP is a particular case of NA. If $p \neq \bot$ then $p \models_\pi p$ (RR). It is enough to apply RM with $p' = p$, $q' = \neg p$ and $r' = \bot$: from $\neg(p \wedge \neg p \models_\pi \bot)$ (this is NA) and $\neg\neg(p \models_\pi p)$ we conclude $\neg(p \models_\pi \bot)$. Q.E.D.

As seen above $p \models_\pi q$ is equivalent to $p \wedge q >_\prod p \wedge \neg q$ where $>_\prod$ is the strict qualitative possibility relation induced by π. Conversely $p >_\prod q$, comes down to a possibilistic entailment relationship.

Lemma 1: $p >_\prod q$ is equivalent to $p \vee q \models_\pi \neg q$.

Proof: Note that the strict preference relation $>_\prod$ verifies the property $p \vee r >_\prod q \vee r \Rightarrow p >_\prod q$. Hence $p >_\prod q$ entails $p \wedge \neg q >_\prod q$ (where $r = p \wedge q$). Letting $\alpha = p \vee q$ and $\beta = \neg q$, $p \wedge \neg q >_\prod q$ reads $\alpha \wedge \beta >_\prod \alpha \wedge \neg\beta$. This is equivalent to $\alpha \models_\pi \beta$, i.e. $p \vee q \models_\pi \neg q$. Conversely, $p \vee q \models_\pi \neg q$ is equivalent to $(p \vee q) \wedge \neg q >_\prod (p \vee q) \wedge q$, i.e. $p \wedge \neg q >_\prod q$ which, along with $p \geq_\prod p \wedge \neg q$ leads to $p >_\prod q$. Q.E.D.

Lemma 1 is closely related to Gärdenfors and Makinson (1992) representation results for non-monotonic inference by means of expectation orderings which correspond to qualitative necessity relations, except that $\top >_C \bot$ is not requested. Conversely it is possible to prove that any non-monotonic inference relation $\vdash$ between propositions in a Boolean algebra satisfying the above properties can derive from a possibility distribution on Ω. Again, this result is closely related to Gärdenfors (1988) representation theorem for belief revision operations in terms of epistemic entrenchment, restated in terms of non-monotonic inference by Gärdenfors and Makinson (1992).

Theorem 1: Given a non-monotonic inference relation $\vdash$ on a finite Boolean algebra (of subsets of Ω), satisfying OR, RR, NA, LLE, RW, AND and RM the relation $>$ defined by

$p > q$ if and only if $p \vee q \vdash \neg q$

is one of qualitative possibility.

Proof: We have to show that relation > satisfies all axioms of a qualitative possibility relation (see Dubois, 1986)

i) Non-triviality: $T > \bot$ expresses that $T \not\vdash T$ (RR)

ii) $p \geq \bot$ or equivalently $\neg(\bot > p)$, i.e. $\neg(p \not\vdash \neg p)$. Indeed assume $p \not\vdash \neg p$ then if $p \neq \bot$, using RR and AND leads to $p \not\vdash \bot$ which contradicts CP. $\bot \geq \bot$ follows from $\neg(\bot \not\vdash p)$, with $p = T$;

iii) Completeness: $p \geq q$ or $q \geq p$, or equivalently $\neg(p > q$ and $q > p)$. It states that we cannot have $p \vee q \not\vdash \neg q$ and $p \vee q \not\vdash \neg p$. Indeed, otherwise using "AND" leads to $p \vee q \not\vdash \neg p \wedge \neg q$ which violates CP again;

iv) Transitivity: $q \geq p$ and $r \geq q$ implies $r \geq p$. It amounts to proving the inconsistency of $\neg(p \vee q \not\vdash \neg q)$, $\neg(q \vee r \not\vdash \neg r)$ and $p \vee r \not\vdash \neg r$. As in Gärdenfors and Makinson (1992) we first prove $p \vee q \vee r \not\vdash \neg r$, applying OR to $p \vee r \not\vdash \neg r$ and $\neg r \wedge q \not\vdash \neg r$; the latter holds from RR and RW provided that $\neg r \wedge q \neq \bot$. If $\neg r \wedge q = \bot$ then $r \vee q = r$ and by LLE applied to $p \vee r \not\vdash \neg r$, $p \vee q \vee r \not\vdash \neg r$. Next, $p \vee q \vee r \not\vdash \neg q \wedge \neg r$ obtains via contraposition of RM applied to $p \vee q \vee r \not\vdash \neg r$ and $\neg(q \vee r \not\vdash \neg r)$ (letting $p' = p \vee q \vee r$, $q' = q \vee r$, and $r' = \neg r$). Lastly RM can again be applied to $p \vee q \vee r \not\vdash \neg q$ (that follows from $p \vee q \vee r \not\vdash \neg q \wedge \neg r$) and $\neg(p \vee q \not\vdash \neg q)$ in the same way, to conclude $p \vee q \vee r \not\vdash \neg p \wedge \neg q$. Combining the results by the AND rule leads to $p \vee q \vee r \not\vdash \neg p \wedge \neg q \wedge \neg r$ which in turn is not compatible with CP;

v) $r \vee p > r \vee q$ implies $p > q$. It amounts to prove that $p \vee q \vee r \not\vdash \neg q \wedge \neg r$ implies $p \vee q \not\vdash \neg q$. First note that $p \vee q \vee r \not\vdash \neg q \wedge \neg r$ is equivalent to $p \vee (q \vee r) \not\vdash \neg(q \vee r)$. This writes $p > r \vee q$. Moreover we have $r \vee q \geq q$ because it expresses that $\neg(r \vee q \not\vdash \neg q \wedge \neg r)$ which is due to CP. The result follows by transitivity.

Q.E.D.

In the finite setting, the existence of a qualitative possibility relation $>_\prod$ on the Boolean algebra 2^Ω ensures the existence of a corresponding possibility distribution π such that $\pi(\omega) > \pi(\omega') \Leftrightarrow p >_\prod q$ where p and q are the propositions whose only models are ω and ω' respectively (Dubois, 1986). Some results in the non-finite case are given in Gärdenfors and Makinson (1992). Lemma 1 ensures that the qualitative possibility relation obtained from Theorem 1 leads to a possibilistic inference that coincides with the original nonmonotonic inference. Clearly the set of 7 postulates in Theorem 1 can be called the basic postulates of possibilistic inference.

4. POSSIBILISTIC LOGIC

A possibilistic knowledge base ($\prod$KB) is a finite set of weighted formulas $B = \{(p_i\ \alpha_i),\ i = 1,n\}$ where α_i represents a lower bound on the degree of necessity $N(p_i)$. The fuzzy set of models of a $\prod$KB has for membership function the least specific possibility distribution π_B which satisfies the set of constraints $N(p_i) \geq \alpha_i$, $i = 1,n$. This possibility distribution exists and is defined by (Dubois, Lang and Prade, 1989)

$$\forall\ \omega \in \Omega,\ \pi_B(\omega) = \min_{i=1,n}\ \{1 - \alpha_i,\ \omega \models \neg p_i\}. \quad (2)$$

This possibility distribution is not necessarily normal and $\sup_{\omega \in \Omega} \pi_B(\omega)$ is called the degree of consistency of the $\prod$KB.

Inference at the syntactic level in possibilistic logic is performed by means of a weighted version of the resolution principle (Dubois and Prade, 1987):

$$\frac{(p \vee q,\ \alpha)\quad (\neg p \vee r,\ \beta)}{(q \vee r\ \ \min(\alpha,\beta))}$$

Proving $(p\ \alpha)$ from a $\prod$KB B comes down to deriving the contradiction $(\bot\ \beta)$ from $B \cup \{(\neg p\ 1)\}$ with a weight $\beta \geq \alpha$. It will be denoted by $B \vdash (p\ \beta)$. We can also compute the degree of inconsistency of B as $\max\{\alpha \mid B \vdash (\bot\ \alpha)\} = \mathrm{Inc}(B)$. This inference method is as efficient as classical logic refutation by resolution, and has been implemented in the form of an A*-like algorithm (Dubois et al., 1987).

More important is the fact that this inference method is sound and complete with respect to the possibilistic semantics of the $\prod$KB (Dubois, Lang and Prade, 1989, 1991b). Namely $B \vdash (p\ \alpha)$ if and only if the degree of consistency of the possibility distribution associated to $B \cup \{(\neg p\ 1)\}$ is less than $1 - \alpha$. Equivalently, $N_B(p) \geq \alpha$ in the sense of π_B.

Preferential possibilistic entailment can also be captured at the syntactical level, and we can prove (Dubois and Prade, 1991b)

$$p \models_\pi q \text{ for } \pi = \pi_B \text{ if and only if } B \cup \{(p\ 1)\} \vdash (q\ \alpha) \text{ with } \alpha > \mathrm{Inc}(B \cup \{(p\ 1)\}).$$

In other words possibilistic logic offers a syntactic inference device for a class of non-monotonic logics and a computerized tool (POSLOG, see Dubois et al., 1991a) for computing non-monotonic inferences whose complexity is similar to classical logic. Clearly possibilistic reasoning copes with partial inconsistency by yielding non-trivial conclusions.

It also corresponds to what Satoh (1990) calls "lazy non-monotonic reasoning" since non-monotonicity appears only in the presence of inconsistency.

5. ENCODING SYSTEM Z IN POSSIBILISTIC LOGIC

A limitation of possibilistic logic is its inability to account for default rules in its language. Namely, it is not possible to express that $N(q \mid p) > 0$ in a ΠKB, in a direct way. Pearl (1990)'s system Z turns out to be very close to the framework of possibility theory but offers one method to handle defaults. Following Pearl (1990), a default rule is of the form $p \rightarrow q$ where $\rightarrow$ is a *non-classical* arrow relating two Boolean propositions. In the whole paper the arrow $\rightarrow$ has this non-classical meaning, the material implication being written as a disjunction. A default rule is interpreted as being verified by a possible world (if $\omega \models p \wedge q$), satisfied (if $\omega \models \neg p \vee q$) or falsified (if $\omega \models p \wedge \neg q$). Thus, default rules are very similar to so-called conditional objects, denoted by $q \mid p$, qualitative counterparts of conditional uncertainty measures such as $Prob(q \mid p)$, which are either true ($\omega \models p \wedge q$), false ($\omega \models p \wedge \neg q$) or inapplicable ($\omega \models \neg p$), and which can be viewed as the set of formulas entailed by $p \wedge q$ and entailing $\neg p \vee q$. The similarity between conditional objects and non-monotonic inference rules has been pointed out (Dubois and Prade, 1991c).

Given a set of default rules $R = \{p_i \rightarrow q_i \mid i = 1,m\}$, Pearl gives a method to rank-order them such that the least specific rules (i.e. with most general antecedents) get the least priority. The ideas is to partition R into an ordered set $\{R_0, R_1, \ldots, R_k\}$ such that rules in R_i are tolerated by all rules in $R_{i+1} \cup \ldots \cup R_k$. A rule $p \rightarrow q$ is tolerated by a set $\{p_i \rightarrow q_i, i = 1,m\}$ if and only if $\{p \wedge q, \neg p_1 \vee q_1, \ldots, \neg p_m \vee q_m\}$ is consistent. From this partition, Pearl attaches to each rule $d \in R_i$ the weight $Z(d) = i \in \mathbb{N}$. This ranking of default rules induces a ranking of possible worlds. The rank $K(\omega)$ of a possible world ω is the rank of the highest-ranked rule falsified by ω, augmented by the unit, i.e.

$$K(\omega) = \max\{Z(d_i) + 1, \omega \models p_i \wedge \neg q_i\} \qquad (3)$$

This ordering of models induces, in turn, an ordering of formulas using a function z such that $z(p) = \min\{K(\omega) \mid \omega \models p\}$, which is a disbelief function in the sense of Spohn(1988). Pearl defines a non-monotonic inference concept denoted $\vdash_1$ such that

$$p \vdash_1 q \Leftrightarrow z(p \wedge q) < z(p \wedge \neg q).$$

The following example illustrates these definitions:

Example

Let R be the following set of defaults

d_1 $b \rightarrow f$ (birds fly),
d_2 $p \rightarrow \neg f$ (penguins do not fly),
d_3 $p \rightarrow b$ (penguins are birds).

It can be easily verified that the rule d_1 is tolerated by d_2 and d_3, since the following interpretation $\{b = 1, f = 1, p = 0\}$ verifies d1 and satisfies both d_2 and d_3. This contrasts with the other rules for which it is not possible to find an interpretation which verifies d_2 (resp. d_3) and satisfies both d_1 and d_3 (resp. d_1 and d_2). So, we have $Z(d_1) = 0$ and $Z(d_2) = Z(d_3) = 1$.

Let us add the facts $\{p, b\}$ to this base, and we are interested to know if $\neg f$ is a plausible consequence of this system. From the system Z we can infer this consequence since $\min_\omega \{K(\omega), \omega \models b \wedge p \wedge \neg f\} = 1$ is lower than $\min_\omega \{K(\omega), \omega \models b \wedge p \wedge f\} = 2$.

The relationship between system Z and possibilistic logic (and its non-monotonic inference $\models_\pi$) is made clear by the following:

Theorem 2: Let π be the possibility distribution built from the model ranking function K, letting

$$\pi(\omega) = 1 - \frac{K(\omega)}{k + 2}$$

where $k = \max_i Z(d_i)$. Then the following results hold:

i) $z(p) = (k + 2)(1 - \Pi(p))$

ii) the world ranking function K computed by system Z using (3) corresponds to the minimally specific possibility distribution that is obtained using (2) from the possibilistic logic program $\{(\neg p_i \vee q_i, \alpha_i), i=1,n\}$ where α_i is obtained from $Z(d_i) = (k + 2)\alpha_i - 1$

iii) $p \models_1 q$ if and only if $p \models_\pi q$.

Proof: By definition we have:

i) $z(p) = \min\{K(\omega), \omega \models p\}$
$= \min\{(k + 2)(1 - \pi(\omega)), \omega \models p\}$
$= (k + 2)[1 - \max\{\pi(\omega), \omega \models p\}]$
$= (k + 2)(1 - \Pi(p))$.

ii) $\pi(\omega) = 1 - \dfrac{K(\omega)}{k + 2}$
$= 1 - \max\left\{\dfrac{Z(d_i) + 1}{k + 2}, \omega \models p_i \wedge \neg q_i\right\}$
using (3).

On the other hand, the least specific possibility distribution that satisfies $N(\neg p_i \vee q_i) \geq \alpha_i$, i = 1,n is, due to (2)

$$\pi'(\omega) = 1 - \max\{\alpha_i, \omega \models p_i \wedge \neg q_i\}$$

then, $\pi = \pi'$ provided that

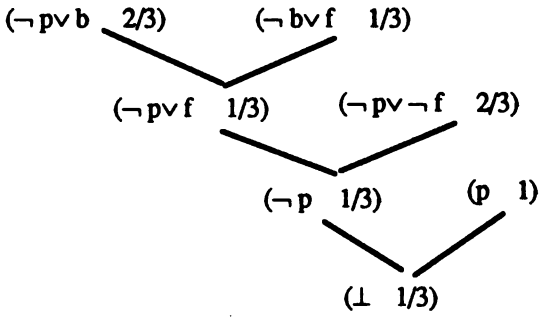
$$\alpha_i = \frac{Z(d_i) + 1}{k + 2}, \; i = 1,n.$$

iii) $p \models_1 q \Leftrightarrow z(p \wedge q) < z(p \wedge \neg q)$
$\Leftrightarrow (k+2)(1 - \Pi(p \wedge q)) < (k+2)(1 - \Pi(p \wedge \neg q))$
$\Leftrightarrow \Pi(p \wedge q) > \Pi(p \wedge \neg q)$
$\Leftrightarrow p \models_\pi q$ Q.E.D

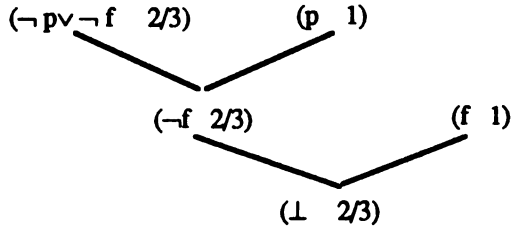
As a consequence, once the default ranking function Z has been computed, the set R of default rules can be represented by a ΠKB: $B = \{(\neg p_i \vee q_i \ \alpha_i), i = 1,n\}$ with $\alpha_i = \frac{Z(d_i) + 1}{k + 2}$ and reasoning with a set of defaults can be achieved by means of the inference machinery of possibilistic logic. Clearly, the minimally specific possibility distribution on interpretations corresponds to the notion of compact ranking by Pearl (1990).

Example (continued)

Notice first that the possibilistic knowledge base Σ equivalent to the set of defaults R constructed from the theorem 1 is $\Sigma = \{(\neg b \vee f \ 1/3), (\neg p \vee b \ 2/3), (\neg p \vee \neg f \ 2/3)\}$. Then the following derivation gives the optimal degree of inconsistency of $\Sigma \cup \{(p\ 1), (b\ 1)\}$ which is equal to 1/3 where:



By refutation, the following derivation shows that $\neg f$ is truly a logical consequence of $\Sigma \cup \{(p\ 1), (b\ 1)\}$ since adding the piece of information (f 1) we find the degree of inconsistency equal to 2/3 which is higher than 1/3:



Theorem 2, and the link between z(p) and Π(p) explains why Boutilier (1991) found that the relation $z(\neg p) \leq z(\neg q)$ is an epistemic entrenchment relation. Note that the two orderings encoded by K and π are oriented in opposite ways. Lastly, Theorem 2 gives a strong justification to a previous hint made by the authors (Dubois and Prade, 1988) about encoding defaults as weighted formulas in the sense of possibility theory.

Remark: It is easy to see that π-entailment (and therefore 1-entailment) overcomes the principal drawback of Adams' logic of conditionals called by (Geffner, 1989; Geffner and Pearl, 1992) irrelevant feature. Indeed, if we have in our knowledge base Σ the rule $(p \rightarrow q \ \alpha)$ then we can conclude $p \wedge r \rightarrow q$ since $\Sigma \cup \{(p\ 1), (r\ 1)\} \models (q\ \alpha)$ with $\alpha > \text{Incons}(\Sigma \cup \{(p\ 1), (r\ 1)\}) = 0$. This result is not surprising since both π-entailment and 1-entailment verify the rational monotony postulate.

6. REASONING WITH PARTIALLY DEFINED POSSIBILITY RELATIONS

While possibilistic logic offers a natural deductive machinery for system Z, the latter has more prima facie expressive power because it enables default rules to be described as such. The priority ranking of defaults is done by system Z while in possibilistic logic, the weights of formulas must be given by the user. Hence the two approaches are complementary. Here we show that in order to endow possibilistic logic with an expressive power equivalent to the Z-system, we must be able to add the qualitative possibility (or necessity) relation to the language.

A qualitative possibilistic knowledge base (QΠKB) is a set of constraints C of the form $\{p_i >_\Pi q_i, i = 1,n\}$ where p_i and q_i are propositional formulae. A weak order $\geq_\pi$ (reflexive, complete, transitive) on Ω is said to be compatible with C if and only if $\forall\ i = 1,n\ \sup\{\omega \mid \omega \models p_i\} >_\pi \sup\{\omega \mid \omega \models q_i\}$. $\geq_\pi$ is called a qualitative possibility distribution (QΠD) on Ω.

Specificity ordering among QΠD's makes sense. Namely any QΠD induces a well-ordered partition of Ω, $\{E_1, \ldots, E_k\}$ such that $\forall\ \omega \in E_i, \forall\ \omega' \in E_j, i < j$, $\omega >_\pi \omega'$ holds. $\geq_\pi$ is said to be less specific than $\geq_{\pi'}$ if and only if the well-ordered partitions $\{E_1, \ldots, E_k\}$ and $\{E'_1, \ldots, E'_{k'}\}$, satisfy

$$\forall\ j = 1, \max(k,k')\ ,\ \bigcup_{i=1,j} E'_i \subseteq \bigcup_{i=1,j} E_i \quad (1)$$

(for $j > \min(k,k')$ we use $E_j = \emptyset$ for $k < k'$). This notion corresponds to the concept of specificity for numerical π's.

Theorem 3: Given two possibility distributions π and π' such that $\pi \leq \pi'$, the ordering $\geq_\pi$ induced from π is more specific than $\geq_{\pi'}$ induced from π'. Conversely, if $\geq_\pi$ is more specific than $\geq_{\pi'}$ then, if we let

$\pi(\omega) = \pi'(\omega')$ if $\omega \in E_j, \omega' \in E'_j$,
$\pi(\omega) > \pi(\omega')$ if $\omega \in E_i, \omega' \in E_j, i < j$,
$\pi'(\omega) > \pi'(\omega')$ if $\omega \in E'_i, \omega' \in E'_j, i < j$,

then $\pi \leq \pi'$.

Proof: $\pi \leq \pi'$ implies (1) is obvious since $\bigcup_{i=1,j} E_i$ corresponds to the level-cuts of π, i.e. are of the form $\{\omega \mid \pi(\omega) \geq \alpha\}$, $\forall\ \alpha \in [0,1]$. As for the converse let $\omega \in E'_j$. Then from (1), $\omega \in E_i$ for $i \leq j$. Hence $\pi'(\omega) = \pi(\omega')$ for any $\omega' \in E_j$ and $\pi'(\omega) = \pi(\omega') \geq \pi(\omega)$. Q.E.D.

Representing a possibility distribution in terms of classes of equally possible worlds, it is clear that the less specific π, the more numerous are the elements in the classes E_j of low rank j. Hence minimizing specificity comes down to minimize the number k of equivalence classes, so as to assign as many worlds as possible to classes of low rank, i.e. to the most compact ranking of worlds, in the terminology of Pearl (1990).

The following theorem establishes the links between system Z and qualitative $\prod$KB's:

Theorem 4: A rule base R in system Z is equivalent to a set of constraints C forming a Q$\prod$KB. Namely each default rule $p_i \rightarrow q_i$ in R can be translated into a constraint $p_i \wedge q_i >_\prod p_i \wedge \neg q_i$ in C. Conversely, any constraint $p_i >_\prod q_i$ in C expresses a default in R of the form $p_i \vee q_i \rightarrow \neg q_i$.

Proof:

- Let us first show that each default in the system Z can be represented by a constraint in C. Indeed, in the system Z from each default $p_i \rightarrow q_i$ we can conclude $p_i \vdash_1 q_i$, then from Theorem 2 $p_i \vdash_1 q_i$ is equivalent to $p_i \wedge q_i >_\prod p_i \wedge \neg q_i$.
- That each constraint $p_i >_\prod q_i$ in C expresses a rule in R of the form $p_i \vee q_i \rightarrow \neg q_i$ is an obvious consequence of Lemma 1 and $\models_\pi = \vdash_1$, as per Theorem 2. Q.E.D.

Any finite consistent Q$\prod$KB induces a partially defined Q$\prod$D $>_\pi$ on Ω, that can be completed according to the principle of minimum specificity. The idea is to try to assign to each world ω the highest possibility level (in forming a well-ordered partition) without violating the constraints. Before giving an algorithm which constructs the well ordered partition of Ω, we transform the set of constraints $C = \{p_i \wedge q_i >_\prod p_i \wedge \neg q_i\}$ obtained from rules $p_i \rightarrow q_i$ into the set of constraints $C' = \{\max(\omega, \omega \models p_i \wedge q_i) > \max(\omega', \omega' \models p_i \wedge \neg q_i)\}$ on models. The following procedure computes the desired partition of Ω:

a. Let E_0 be the empty set.
b. While Ω is not empty repeat b.1.-b.3.:
  b.1. Put in E_i every model which does not appear in the right side of any constraints of C',
  b.2. Remove the elements of E_i from Ω,
  b.3. Remove from C any constraint containing elements of E_i.

Example:

Let R be the set of the following defaults $R = \{d_1: b \rightarrow f, d_2: p \rightarrow \neg f, d_3: p \rightarrow b\}$, which is translated into the following set C of constraints:

$$b \wedge f >_\prod b \wedge \neg f$$
$$p \wedge \neg f >_\prod p \wedge f$$
$$p \wedge b >_\prod p \wedge \neg b$$

And let Ω be the following set of possible models $\Omega = \{\omega_0: \neg b \wedge \neg f \wedge \neg p,\ \omega_1: \neg b \wedge \neg f \wedge p,\ \omega_2: \neg b \wedge f \wedge \neg p,\ \omega_3: \neg b \wedge f \wedge p,\ \omega_4: b \wedge \neg f \wedge \neg p,\ \omega_5: b \wedge \neg f \wedge p,\ \omega_6: b \wedge f \wedge \neg p,\ \omega_7: b \wedge f \wedge p\}$, Then the set of constraints C' on models is:

C'_1: $\max(\omega_6,\omega_7) >_\prod \max(\omega_4,\omega_5)$
C'_2: $\max(\omega_5,\omega_1) >_\prod \max(\omega_3,\omega_7)$
C'_3: $\max(\omega_5,\omega_7) >_\prod \max(\omega_1,\omega_3)$

Let us apply now the previous algorithm, the models which do not appear in the right side of any constraint of C' are $\{\omega_0,\omega_2,\omega_6\}$, we call this set E_0. We remove the elements of E_0 from Ω and we remove the constraint C'_1 from C' (since assigning to ω_6 the highest possibility level makes the constraint C'_1 always satisfied). We start again the procedure and we find successively the two following sets $\{\omega_4,\omega_5\}$ and $\{\omega_1,\omega_3,\omega_7\}$. Finally, the well ordered partition of Ω is:

$$\{\omega_0=\omega_2=\omega_6\} >_\prod \{\omega_4=\omega_5\} >_\prod \{\omega_1=\omega_3=\omega_7\}.$$

Note that for instance it is possible to apply rules of inference such as

$$(p \wedge q >_\prod p \wedge \neg q \text{ and } p \wedge r >_\prod p \wedge \neg r) \text{ implies}$$
$$p \wedge q \wedge r >_\prod p \wedge q \wedge \neg r \text{ and } p \wedge q \wedge r >_\prod p \wedge \neg q \wedge r$$

to go down from formulas to models. Indeed, the two first premisses are equivalent to $p >_\prod p \wedge \neg q$ and $p >_\prod p \wedge \neg r$. Hence, $p =_\prod (p \wedge \neg q) \vee (p \wedge q) =_\prod (p \wedge q)$. Finally, $p \wedge q >_\prod p \wedge \neg r \geq_\prod p \wedge \neg r \wedge q$; in other words we have $p \wedge q \wedge r >_\prod p \wedge q \wedge \neg r$.

We can prove that the ranking of models obtained by the above algorithm is the same as the one computed by Pearl (1990) from his rule-ranking algorithm.

Theorem 5: Given a set R of defaults $\{p_i \rightarrow q_i,\ i = 1,n\}$, and the corresponding Q$\prod$KB inducing a minimally specific Q$\prod$D $\geq_\pi$, then the rank ordering function K on models, as defined in (3) is such that

$$K(\omega) \leq K(\omega') \text{ if and only if } \omega \geq_\pi \omega'.$$

Proof: Consider $C = \{p_i \wedge q_i >_{\prod} p_i \wedge \neg q_i, i = 1,n\}$. Consider $\omega \in E_0$; ω is such that $\forall\, i = 1,n, \neg(\omega \models p_i \wedge \neg q_i)$. And ω is maximal for $\geq_\pi$. Then $K(\omega) = 0$ since the maximizing set in (3) is empty. Now let C_0 be the set of constraints such that $\omega \models p_i \wedge q_i$ for some $\omega \in E_0$. Clearly any constraint i in C_0 is such that for some $\omega \in E_0$, $\omega \models \{p_i \wedge q_i\} \cup \{\neg p_j \vee q_j, j \neq i\}$. Hence default d_i is tolerated by the others. The converse is obvious. Hence the set of constraints C_0 removed at step b.3 corresponds to the set of defaults R_0 such that $Z(d_i) = 0$. The next run of step b is thus applied to the constraints corresponding to $R\text{-}R_0$. The models found next by our algorithm would thus have $K(\omega) = 1$ using Pearl's procedure and would be maximal for $\geq_\pi$ on $\Omega\text{-}E_0$ and strictly dominated by E_0. A simple recursion can end the proof.

Q.E.D.

Concluding, the well-ordered partitions corresponding to the minimally specific Q$\prod$D built from C and to the ranking function (called "compact" by Pearl) of models obtained by the system Z procedure are the same.

Remark

Pearl (1990) also considers a less adventurous consequence relation $\vdash_0$ such that $p \vdash_0 q$ if and only if the set $R \cup \{p \rightarrow \neg q\}$ is inconsistent. The corresponding consequence relation in Q$\prod$KB's has been recently studied by Fariñas del Cerro et al. (1992). Namely, they have considered the consequence relation $\models_\forall$ such that $p \models_\forall q$ if and only if $p \wedge q >_{\prod} p \wedge \neg q$ for all qualitative possibility relations $>_{\prod}$ induced by a feasible Q$\prod$D $\geq_\pi$ with respect to C.

It should be pointed out that $p \models_\forall q$ is equivalent to $p \vdash_0 q$. Indeed observe that $p \models_\forall q$ requires the inconsistency of $C \cup \{p \wedge \neg q >_{\prod} p \wedge q\}$. Besides Pearl (1990) has shown that $p \vdash_0 q$ guarantees that $p \wedge q >_{\prod} p \wedge \neg q$ to be valid in all qualitative possibility relations $>_{\prod}$ induced by a feasible Q$\prod$D $\geq_\pi$ with respect to C (Pearl stated this in terms of K function, namely "0-entailment guarantees that $z(p \wedge q) < z(p \wedge \neg q)$ holds in *all* admissible ranking functions K"; indeed each Q$\prod$D was called feasible admissible ranking in (Pearl, 1990)).

The problem of reasoning with Q$\prod$KB's at the syntactic level without resorting to a preprocessing of defaults, remains open, although inference rules can be built as suggested above. Moreover Q$\prod$KB's are equivalent to numerical $\prod$KB's only with respect to the consequence relation $\models_\pi$. We have no access to consequence relation $\models_\forall$ in a numerical $\prod$KB's because the set of Q$\prod$D's more specific than the least specific Q$\prod$D with respect to C is larger than the set of feasible Q$\prod$D's.

7. POTENTIAL INCONSISTENCY AND DEFAULT RULES

The definition of "tolerance" given by Pearl, can be interpreted in the following way. A default d is said to be tolerated if and only if putting its antecedent side to true we preserve the consistency of the knowledge base. Indeed, if a default d: $p \rightarrow q$ is tolerated then by definition there exists a world which makes p and q true (verifies d) and satisfies the other defaults, therefore when the antecedent side of d (p) is true the knowledge base is still consistent (the converse can be shown in the similar manner). Tolerance checking is thus related to the problem of computing, for a given classical knowledge base Σ, the set of formulas, built from special literals called "inputs", which make it inconsistent when added. This set will be denoted by P-Incons (Potential Inconsistencies). The interest of detecting potential inconsistency has been stressed by Yager and Larsen (1991).

Let Σ be a classical knowledge base built from the set of defaults R by changing $p \rightarrow q$ into $\neg p \vee q$. Ante(d) denotes the set of literals in the antecedent side of a default d. We suppose that Ante(d) is a conjunction of literals. E denotes the set of all such literals or their negation, taken as inputs to Σ. Let P-Incons = $\{A, A \cup \Sigma \vdash \perp$ and $A \subset E\}$. Once P-Incons is computed, deciding if a given default d is tolerated requires only to check if its antecedent side does not contain any element of P-Incons:

Lemma 2: $\forall\, d \in R$, d is tolerated by R-{d} iff $\nexists\, A \in$ P-Incons such that $A \subseteq$ Ante(d).

Proof: Indeed, suppose that d is tolerated; in other words if Ante(d) is true then the knowledge base Σ will be still consistent and therefore we cannot deduce the contradiction from Ante(d) $\cup\, \Sigma$ nor from any subset of Ante(d), and therefore there does not exist a subset of Ante(d) which belongs to P-Incons. The second part of the proof goes in a similar way. Q.E.D.

Clearly the notion of tolerated default is thus closely linked to the discovery of potential inconsistencies in classical knowledge bases caused by the arrival of inputs. When the antecedent side of d can be, as in (Pearl, 1990), any formula, Ante(d) represents the set of configurations of literals which make the antecedent side of d true, and if there are several configurations then it is enough to find only one which does not contain any element of P-Incons for the default d to be tolerated.

For the need of the application (see the next paragraph) the structure of P-incons is enriched to contain pairs (A Neg(A)) rather than only sets A, where Neg(A) contains the "label" of $\neg$A in the terminology of ATMS (De Kleer, 1986), viewing defaults as assumptions. Neg(A) contains the minimal subsets of formulas in Σ that contradict A. They correspond to all the minimal

subsets of defaults in R which have allowed us to deduce $\neg A$. Clearly this can be done by means of an ATMS if we associate to each clause C_i of the knowledge base Σ a specific assumption which controls C_i; namely the symbol d_i represents default d_i and appears every time, C_i is included in some resolution step. The set P-Incons must be minimal, namely it must not contain two elements (A Neg(A)) and (A' Neg(A')) such that $A \subseteq A'$ (then only A is kept).

More precisely, we construct first a knowledge base $\Sigma = \{C_i\}$ from the set of defaults $R = \{p_i \rightarrow q_i\}$ where C_i represents the clausal form of $p_i \rightarrow q_i$, and then we change Σ into $\Sigma' = \{(\neg d_i \vee C_i)\}$ and finally we compute, using the ATMS, the nogoods of Σ', which are of the form:

Nogood = $\{X \cup A / X \cup A \cup \Sigma' \vdash \perp$, where X contains assumptions d_i only and A input literals$\}$.

And consequently: (A, Neg(A)) $\in$ P-Incons means $A \subset E$, $A \cup X \in$ Nogood and Neg(A) = {X, $A \cup X \in$ Nogood}.

Another use of the ATMS is in checking the inconsistency of a set of defaults as we see it in the following proposition:

Theorem 6: If $\exists$(A, Neg(A)) $\in$ P-Incons and $\exists$ X $\in$ Neg(A) such that $\forall$ d $\in$ X, $A \subseteq$ Ante(d) then the knowledge base Σ is inconsistent.

Proof: Indeed, let X correspond to a set of defaults where the antecedent side of every default contains a set of literals A such that $A \cup \Sigma \vdash \perp$. Then no default of X is tolerated (from Lemma 2), and therefore the set X is inconsistent and by consequence the knowledge base is also inconsistent. (X is called a Clash in (Geffner, 1989)). Q.E.D

These results indicate the usefulness of an ATMS for implementing a default ranking procedure.

8. FREE DEFAULTS AND THE INHERITANCE PROBLEM

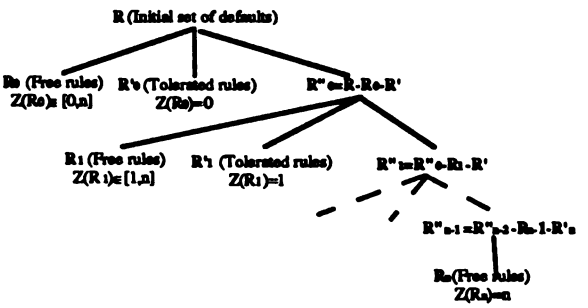
The principal drawback of the system Z is its inability to solve the problem of the blocking of property inheritance. For example, from the set of defaults R = {d1: b $\rightarrow$ f, d2: p $\rightarrow$ b, d3: p $\rightarrow \neg$f, d4: b $\rightarrow$ w} where b = bird, p = penguin, f = fly, w = wings, the inference $p \vdash_1 w$ (penguins have wings) is not valid, while the only undesirable property for the penguin is "flying". In other words the only conflict between the general class (that of birds) and the specific class (that of penguins) concerns the property "flying" and the characteristic "having wings" should be outside the conflict.

This problem has been addressed by Geffner (1989) using a partial ordering of defaults, which makes the inference difficult to handle at a practical level.

We propose here to set aside defaults that are not involved in any potential inconsistency of the knowledge base Σ attached to a set of defaults R. These defaults will be called free defaults and they form the set Free(R). Notice that a free default always has a minimal Z-rank (the lowest priority) regardless of the way it is written, the converse is not true; indeed in the last example the defaults b $\rightarrow$ f and b $\rightarrow$ w have both the lowest priority but only the default b $\rightarrow$ w is free. The set Free(R) can be computed from P-incons as the set of defaults that do not belong to any set X in any Neg(A) for any (A Neg(A)) $\in$ P-Incons, i.e.

Free(R) =
$\{d \mid d \notin \cup_i [\text{Neg}(A_i)]$ where $(A_i \text{ Neg}(A_i) \in$ P-Incons$\}$

When translating Z-ranked defaults into possibilistic clauses, we propose here to assign interval-valued weights to the corresponding free clauses. The value associated to free defaults is itself free and will then be chosen in the interval so as to derive appropriate conclusions. The following hierarchy shows how the set of defaults is partitioned and summarizes the values assigned to each default:



In the above example, the default d4: b $\rightarrow$ w is free and we assign to it the value "$\geq$ 0" then the inference $p \vdash_1 w$ becomes valid because we are then allowed to raise the Z-value up to 2. It is then possible to use possibilistic logic again in order to compute this inference.

Example

Let R be the following set of defaults:

d1: na $\rightarrow$ ce	(nautilus is a cephalopod)
d2: ce $\rightarrow$ mo	(cephalopod is a mollusc)
d3: na $\rightarrow$ co	(nautilus have shell)
d4: mo $\rightarrow$ co	(mollusc have shell)
d5: ce $\rightarrow \neg$co	(cephalopod have no shell)

Notice that with system Z the relation $na \vdash_1 mo$ is not valid. d4 is the only tolerated default by R and then Z(d4) = 0. There are no free defaults.

When we remove d_4 from R, the defaults d_2 becomes free which means that $Z(d_2) \geq 1$ (for example $Z(d_2) = 2$). The defaults d_5 is tolerated by $R-\{d_2,d_4\}$ then $Z(d_5) = 1$. And finally, $Z(d_3) = Z(d_1) = 2$. With these assignments, the inference na $\vdash_1$ mo becomes valid.

In this section we have proposed a default ranking procedure which overcomes some drawbacks of system Z. But this algorithm still requires some refinements. Indeed, if we add to the set of defaults $R = \{d_1: b \rightarrow f, d_2: p \rightarrow b, d_3: p \rightarrow \neg f\}$ the defaults $\{d_4: lb \rightarrow b, d_5: b \rightarrow l; d_6: lb \rightarrow \neg l\}$ where "lb" means legless-bird and l "l" means legs, then the relations $p \vdash_1 \neg l$ and $lb \vdash_1 f$ cannot be infered because $\{d_1: b \rightarrow f, d_5: b \rightarrow l\}$ are not free. We can refine the Z-ranking, by strictly ordering defaults which belong to the same set of tolerated defaults. This refinement, similar to what is done in the Z^+ system (Goldszmidt and Pearl, 1991b) is not satisfactory because the defaults to which a low priority is given will be cancelled by the others even if they are not logically related to them.

In other to overcome this problem, we slightly change the definition of Free defaults to be a set of defaults which are not involved in any inconsistency (rather than in any potential inconsistency) of the knowledge base Σ. In other words, the set E (inputs of Σ) is restricted to be the set of observed facts. The following schema (Fig. 1) summarizes the procedure of infering plausible conclusions:

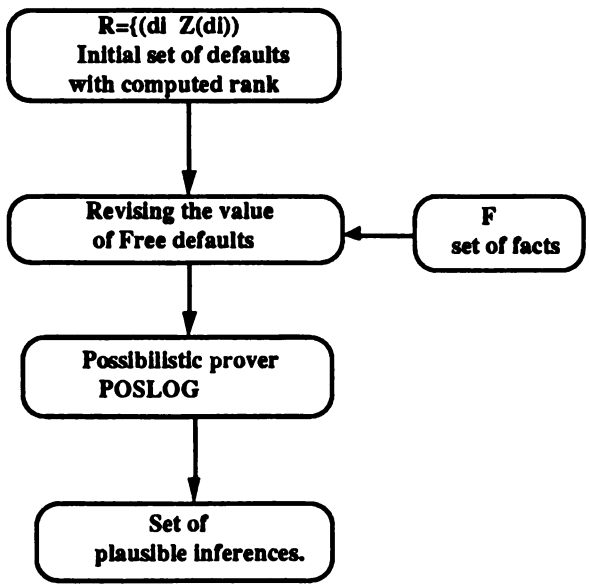


Figure 1

Example

Let R be the previous set of defaults:

$R = \{d_1: b \rightarrow f, d_2: p \rightarrow b, d_3: p \rightarrow \neg f, d_4: lb \rightarrow b, d_5: b \rightarrow l; d_6: lb \rightarrow \neg l\}$

The system Z assigns the value 0 to the defaults d_1 and d_5 and the value 1 to the other defaults.
Let $F = \{p\}$ be the set of facts, and let Σ the knowledge base built from R by changing each default $p \rightarrow q$ by the corresponding clause $\neg p \vee q$.
It is easy to check that $\Sigma \cup F$ is inconsistent. The minimal set of clauses which is responsible of the inconsistency of $\Sigma \cup F$ is composed of the clauses corresponding to the defaults d1, d2 and d3 and obviously the fact p. Then, the set of Free defaults is:

$$Free(R) = \{d3, d4, d5\}$$

It is easy to check if we do not revise the ranking of the free defaults then the inference $p \vdash_1 lb$ will not be valid, while revising values of the free defaults to be 1 the desired inference holds (the new value of free defaults must be at least larger then the degree of inconsistency of the incoherent knowledge base; in our case the degree of inconsistency is equal to 0 (in the sense of Pearl)).

Still another way of solving the problem of the blocking of property inheritance is to properly use the possibilistic knowledge base Σ equivalent to a Z-ranked set of defaults (where free defaults are distinguished). Indeed a given input will create a partial inconsistency and all formulas with a weight lower than the inconsistency level are cancelled by possibilistic resolution. This is where the problems come from. Then it makes sense to inhibit only those formulas whose weight is equal to the inconsistency level and restore consistency without deleting the other formulae. There may be several ones to delete since there may be several inconsistencies. But if here are no two defaults which have the same weight, this way of proceeding will correspond to select a particular maximally consistent knowledge sub-base, the "most entrenched" one. In this way the above suggested refinement procedure then no longer creates a problem. The weight becomes a "tag" to analyze inconsistencies. Dubois, Lang and Prade (1990) propose some solutions to the problem of selecting the best maximal consistent sub-bases of an inconsistent knowledge base. One of the solutions was shown to be equivalent to what (Brewka, 1989) calls "Preferred subtheories" (see Benferhat et al., 1992). Moreover Boutilier in a recent paper (Boutilier, 1992) uses Brewka's preferred subtheories in the system Z to define a new nonmonotonic inference relation. The idea is to view the set of defaults with computed rank as a layered set of defaults $R = R_n \cup R_{n-1} \cup \ldots \cup R_0$ where R_i contains only rules of weight equal to i. A preferred subtheory D of R is constructed from the set of observed facts F and adding as many formulas of the set R_n as possible (with respect to consistency criterion) then as many as possible formulas of the set R_{n-1}, and so on. Lastly, Boutilier defines the set of plausible conclusions

as a set of assertions which hold in all preferred subtheories of R.

9. CONCLUSIONS AND FUTURE WORKS

We have seen how possibilistic logic can be used to implement the bold inference of system Z, and that the notion of qualitative possibility relation can be used to model default rules. Strict defaults of Goldszmidt and Pearl (1991a) can also be accounted for. Namely $p \Rightarrow q$ (expressing $Prob(q \mid p) = 1$) can be handled as $\bot \geq_{\prod} p \wedge \neg q$ and $p \wedge q >_{\prod} p \wedge \neg q$. These results are one more step to promote a unified view of uncertainty management and non-monotonic reasoning that seems to emerge presently. An interesting subsequent issue is to relate Pearl's results on default reasoning to the nonmonotonic logic of conditional objects (Dubois and Prade 1991c). Further research will also focus on the relationship between system Z^+ and possibility theory. Namely, using the links between the Z^+ system and Spohn's ordinal conditional functions (Spohn, 1988) on the one hand, and the links between the latter and possibility theory (Dubois and Prade, 1991a), we can prove that the notion of conditional possibility at work in system Z^+ derives from Dempster rule of conditioning (Shafer, 1976), turning min into product in (1). We may then express a default $p \xrightarrow{\delta} q$ with strength δ in system Z+ as an inequality

$$k \cdot \prod(p \wedge q) > \prod(p \wedge \neg q)$$

where $k \in (0,1)$ is a function of the integer $\delta \in \mathbb{N}$.

References

E.W. Adams (1975). *The Logic of Conditionals.* Dordrecht: D. Reidel.

S. Benferhat, D. Dubois, J. Lang, and H. Prade (1992). Hypothetical reasoning in possibilistic logic: basic notions and implementation issues. In P.-Z. Wang and K.F. Loe (eds.), *Advances in Fuzzy Systems: Applications and Theory - Vol. 1*, Singapore: World Scientific Publishing Comp..

C. Boutilier (1991). Default priorities as epistemic entrenchment. Tech. Report KRR-TR-91-2, Computer Science Department, University of Toronto, Ontario, 140-147.

C. Boutilier (1992). What is a Default priority?. *Proc. of the 9th Canadian Conv. on Artificial Intelligence (AI'92)*, Vancouver, May 1992, 140-147.

G. Brewka (1989). Preferred subtheories: an extended logical framework for default reasoning. *Proc. of the Inter. Joint Conf. on Artificial Intelligence (IJCAI'89)*, Detroit, 1043-1048.

J. De Kleer (1986). An assumption-based TMS. Artificial Intelligence, 28, 127-162; "Extending the ATMS", Artificial Intelligence, 28, 163-196.

Dubois D. (1986) Belief structures, possibility theory, decomposable confidence measures on finite sets. *Computer and Artificial Intelligence* 5(5):403-417.

D. Dubois, J. Lang, and H. Prade (1987). Theorem-proving under uncertainty – A possibility theory-based approach. *Proc. of the 10th Inter. Joint Conf. on Artificial Intelligence (IJCAI-87)*, Milano, Italy, 984-986.

D. Dubois, J. Lang, and H. Prade (1989). Automated reasoning using possibilistic logic: semantics, belief revision and variable certainty weights. *Proc. of the 5th Workshop on Uncertainty in Artificial Intelligence*, Windsor, Ontario, 81-87. Revised version in *IEEE Trans. on Data and Knowledge Engineering*, to appear.

D. Dubois, J. Lang, and H. Prade (1990). Inconsistency in possibilistic knowledge bases – To live or not live with it. In Tech. Report IRIT/90-54/R, I.R.I.T., Université Paul Sabatier, Toulouse, France. To appear in L.A. Zadeh and J. Kacprzyk (eds.), *Fuzzy Logic for the Management of Uncertainty*, Wiley, 1992.

D. Dubois, J. Lang, and H. Prade (1991a). Towards possibilistic logic programming. *Proc. of the Inter. Conf. on Logic Programming'91* (K. Furukawa, ed.), Paris, June 25-28, Cambridge, Mass.: The MIT Press, 581-595.

D. Dubois, J. Lang, and H. Prade (1991b). Possibilistic logic. In Tech. Report IRIT/91-98/R, I.R.I.T., Université Paul Sabatier, Toulouse, France. To appear in D.M. Gabbay (ed.), *Handbook of Logic in Artificial Intelligence and Logic Programming*, Vol. 3, Oxford University Press, 1992.

D. Dubois, and H. Prade (1987). Necessity measures and the resolution principle. *IEEE Trans. on Systems, Man and Cybernetics* 17:474-478.

D. Dubois, and H. Prade (1988). Default reasoning and possibility theory. Artificial Intelligence, 35, 243-457.

D. Dubois, and H. Prade (1990). The logical view of conditioning and its application to possibility and evidence theories. *Int. J. of Approximate Reasoning* 4: 23-46.

Dubois D., Prade H. (1991a) Epistemic entrenchment and possibilistic logic. *Artificial Intelligence* 50:223-239.

D. Dubois, and H. Prade (1991b). Possibilistic logic, preferential models, non-monotonicity and related issues. *Proc. of the 12th Inter. Joint Conf. on Artificial Intelligence (IJCAI-91)*, Sydney, Australia, Aug. 24-30, 419-424.

D. Dubois, and H. Prade (1991c). Conditional Objects and Non-monotonic Reasoning. *Proc. of the 2nd Inter. Conf.*

on Principles of Knowledge Representation and Reasoning (KR'91) (J. Allen, R. Fikes and E Sandewall, eds.), Cambridge, MA: Morgan-Kaufmann, April 22, 175-785.

D. Dubois, and H. Prade (with the collaboration of H. Farreny, R. Martin-Clouaire R., and C. Testemale) (1988). *Possibility Theory – An Approach to Computerized Processing of Uncertainty*. New York: Plenum Press.

L. Fariñas del Cerro, A. Herzig, and J. Lang (1992). From expectation-based nonmonotonic reasoning to conditional logics. *Working Notes of the 4th Inter. Workshop on Nonmonotonic Reasoning*, Plymouth, Vermont, May 28-31, 79-86. Short version in *Proc. 10th Europ. Conf. on Artificial Intelligence (ECAI'92)*, Vienna, Austria, Aug. 3-7, 1992, to appear.

D.M. Gabbay (1985). Theoretical foundations for non-monotonic reasoning in expert systems. In K.R. Apt (ed.), *Logics and Models of Concurrent Systems*, Berlin: Springer Verlag, 439-457.

P. Gärdenfors (1988). *Knowledge in Flux – Modeling the Dynamic of Epistemic States*. Cambridge, Mass.: The MIT Press.

P. Gärdenfors (1991). Nonmonotonic inferences based on expectations: a preliminary report. *Proc. of the 2nd Inter. Conf. on Principles of Knowledge Representation and Reasoning (KR'91)* (J. Allen, R. Fikes and E. Sandewall, eds.), Cambridge, Mass., April 22-25, Morgan & Kaufmann, 585-590.

P. Gärdenfors, and D. Makinson (1992). Non-monotonic inference based on expectations. *Artificial Intelligence*, to appear.

H. Geffner (1989). Default reasoning: causal and conditional theories. Tech. Rep. 137, Cognitive Systems Laboratory, Department of Computer Science, UCLA, Los Angeles, CA.

H. Geffner, and J. Pearl (1992). Conditional entailment: bridging two approaches to default reasoning. *Artificial Intelligence* **53**:209-244.

M. Goldszmidt, and J. Pearl (1991a). On the consistency of defeasible databases. *Artificial Intelligence* **52**:121-149.

M. Goldszmidt, and J. Pearl (1991b). System-Z^+: a formalism for reasoning with variable-strength defaults. *Proc. of the 9th National Conf. on Artificial Intelligence (AAAI-91)*, 16-19 July, 399-404.

S. Kraus, D. Lehmann, and M. Magidor (1990). Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence* **44**:167-207.

D. Makinson (1989). General theory of cumulative inference. In M. Reinfrank, J. De Kleer, M.L. Ginsberg and E. Sandewall (eds.), *Non-Monotonic Reasoning* (Proc. of the 2nd Inter. Workshop, Grassau, FRG, June 1988), LNAI Vol. 346, Berlin: Springer Verlag, 1-18.

D. Makinson, and P. Gärdenfors (1991). Relations between the logic of theory change and nonmonotonic logic. In A. Fuhrmann, M. Morreau (eds.), *The Logic of Theory Change* (Proc. of the Workshop, Konstanz, FRG, Oct. 1989), LNAI Vol. 465, Berlin: Springer Verlag, 185-205.

J. Pearl (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann Publ. Inc..

J. Pearl (1990). System Z: a natural ordering of defaults with tractable applications to default reasoning. *Proc. of the 3rd. Conf. on Theoretical Aspects of Reasoning About Knowledge* (M. Vardi, ed.), Morgan Kaufman, San Mateo, CA, 121-135.

D. Satoh (1990). A probabilistic interpretation for lazy nonmonotonic reasoning. *Proc. of the 8th National Conf. on Artificial Intelligence (AAAI'90)*, July 29-Aug. 3, 659-664.

G. Shafer (1976). *A Mathematical Theory of Evidence*. New Jersey: Princeton University Press.

Y. Shoham (1988). *Reasoning About Change – Time and Causation from the Standpoint of Artificial Intelligence*. Cambridge, Mass.: The MIT Press.

W. Spohn (1988). Ordinal conditional functions: a dynamic theory of epistemic states. In W.L. Harper and B. Skyrms (eds.), *Causation in Decision, Belief Change, and Statistics – Vol. II*, Dordrecht: Kluwer Academic Publ., 105-134.

R.R. Yager, Larsen H.L. (1991). On discovering potential inconsistencies in validating uncertain knowledge bases by reflecting on the input. *IEEE Trans. on Systems, Man and Cybernetics* **21**:790-801.

L.A. Zadeh (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems* **1**:3-28.

Normative, Subjunctive and Autoepistemic Defaults: Adopting the Ramsey Test

Craig Boutilier
Department of Computer Science
University of British Columbia
Vancouver, British Columbia
CANADA, V6T 1Z2
email: cebly@cs.ubc.ca

Abstract

We explore the relationship between theories of nonmonotonic inference and belief revision using modal and conditional representations. We demonstrate that the logics governing belief revision and conditional default reasoning are identical by showing that the *normative* and *subjunctive* conditionals, defined in a modal logic CO* in earlier work, have precisely the same formal truth conditions. This can only be achieved by a strict adherence to the *Ramsey test* in our formulations. Hence, we must take pains to avoid the celebrated *triviality results*. We explain this formal similarity, the practical considerations that distinguish the two types of reasoning, and how default reasoning can be viewed as a special case of belief revision. We then show that autoepistemic logic can be generalized within this framework and discuss the relationship between autoepistemic, normative and subjunctive defaults.

1 Introduction

Default reasoning is the process of drawing conclusions in the face of incomplete information or based on premises that allow exceptions. The defeasible nature of default reasoning undermines any guarantee that these conclusions are true, and new evidence may force a change in the belief of such. For this reason the process is nonmonotonic. Belief revision focuses on changing beliefs that are mistaken, and how to include or discount new information in a principled manner. Since default conclusions can be mistaken, the two types of reasoning complement each other quite nicely, a default theory allowing one to make mistakes and a theory of revision specifying how such mistakes are to be corrected; but they also seem somewhat distinct and the study of these problems has traditionally remained somewhat isolated.

Some researchers have insisted that they have a much deeper connection than this simple symbiotic relationship. This connection is implicit in the recent approaches to default reasoning using "normative" conditional logics (e.g. (Boutilier 1990; Boutilier 1991; Kraus, Lehmann and Magidor 1990; Lehmann 1989; Delgrande 1988)), all of which appear to be more or less the same (Boutilier 1990; Kraus, Lehmann and Magidor 1990; Pearl 1990). These logics share certain properties with counterfactual and subjunctive logics (Lewis 1973; Ginsberg 1986), which can be viewed via the *Ramsey test* to capture a form of belief revision (Gärdenfors 1988). But differences remain: "[subjunctive] conditional logic refers implicitly to the *actual* state of the world whereas [normative conditionals] do not" (Kraus, Lehmann and Magidor 1990, p.170). However, as Lehmann has observed, we should not be surprised if the two logics turn out to be the same. The connection between belief revision and nonmonotonic logic has been explored in detail by Gärdenfors and Makinson (1990; 1991), where these types of reasoning are viewed as "two sides of the same coin." They describe how the logic of theory change can be used to characterize nonmonotonic consequence relations, and how conditions on revision operators correspond naturally to conditions on nonmonotonic logics.

We pursue this connection further here. We will explore the differences between default reasoning and belief revision, claiming that this dichotomy reflects primarily pragmatic considerations that distinguish the two. In particular, the formal structure of the two processes is identical. This connection is developed within a uniform semantic framework based on a bimodal logic CO*. To get started we quickly review CO*, within which we have recently developed a *subjunctive conditional* $\xrightarrow{KB}$. $A \xrightarrow{KB} B$ is read "revising (an agent's implicit belief set) KB by A results in belief in B" (Boutilier 1992d). We have shown that this approach satisfies the *AGM postulates* for revision (Section 2.2). We also examine the *normative conditional* $\Rightarrow$ that has been defined in CO* in earlier work (Boutilier 1990; Boutilier 1991) for default reasoning (Section 2.3). We read $A \Rightarrow B$ "A normally implies B" and treat these as default rules. These have been shown to capture a number of approaches to default reasoning, including (Pearl 1988; Pearl 1990; Lehmann 1989).

To demonstrate the equivalence of revision and default reasoning we proceed in two stages. First we discuss the truth conditions for $\xrightarrow{KB}$ and show that it respects the Ramsey test.

This stands in apparent contradiction with the Gärdenfors (1986; 1988) *triviality result*, which states that no subjunctive conditional can be defined in terms of the Ramsey test and respect the AGM postulates. We argue that once we permit conditionals (and beliefs) in our belief sets the key postulate of AGM revision must be weakened somewhat. Our adherence to the Ramsey test and the expressive power of CO* distinguishes our subjunctive logic from existing logics and provides us with some rather different patterns of inference. It is precisely the patterns of inference we reject for the connective $\xrightarrow{KB}$ that have traditionally distinguished default reasoning from revision. Second, we compare the normative and subjunctive conditionals. What is surprising is the fact that $\Rightarrow$ and $\xrightarrow{KB}$ have precisely the same definition in CO*, or the same formal truth conditions. The conditionals and logics underlying default reasoning and belief revision are therefore identical.

Finally, we examine the practical considerations that distinguish the two types of reasoning and argue that default reasoning can be viewed as the revision of a *theory of expectations*. Since our approach to revision relies heavily on the ability of CO* to represent "only knowing," we can also show that CO* generalizes autoepistemic logic. This has certain implications for the distinction between autoepistemic and normative or subjunctive defaults. It also demonstrates the importance of allowing beliefs and conditionals in belief sets.

2 Normative and Subjunctive Conditionals

2.1 The Modal Logic CO*

The semantic framework we adopt for the unification of default reasoning and belief revision is a standard Kripkean possible worlds model. However, we use an extended bimodal language, allowing considerably increased expressive power compared to the usual modal language with a single modal operator. In this section, we review the bimodal logic CO* and define a normative and subjunctive conditional within the logic. The presentation is brief and we refer to (Boutilier 1990; Boutilier 1991) and to (Boutilier 1992d) for further details and motivation.

The modal logic CO is based on a propositional language (over variables P) augmented with two modal operators $\Box$ and $\breve{\Box}$. $\mathbf{L}_{CPL}$ denotes the propositional sublanguage of this bimodal language $\mathbf{L}_B$. The sentence $\Box\alpha$ is read in the standard way as "α is true at all *accessible* worlds." In contrast, $\breve{\Box}\alpha$ is read "α is true at all *inaccessible* worlds." A CO-model is a triple $M = \langle W, R, \varphi\rangle$, where W is a set of worlds with valuation function φ and R is an accessibility relation over W. We insist that R be transitive and connected.[1] We note that CO-structures consist of a totally-ordered set of *clusters* of worlds, where a cluster is simply a *maximal* set of worlds $C \subseteq W$ such that wRv for each $w, v \in C$ (that is, no extension of C enjoys this property). This is evident in Figure 1, where each large circle represents a cluster of mutually accessible worlds.

Satisfaction of a modal formula at world w is given by:

1. $M \models_w \Box\alpha$ iff for each v such that wRv, $M \models_v \alpha$.
2. $M \models_w \breve{\Box}\alpha$ iff for each v such that not wRv, $M \models_v \alpha$.

We define several new connectives as follows: $\Diamond\alpha \equiv_{df} \neg\Box\neg\alpha$; $\breve{\Diamond}\alpha \equiv_{df} \neg\breve{\Box}\neg\alpha$; $\overline{\Box}\alpha \equiv_{df} \Box\alpha \wedge \breve{\Box}\alpha$; and $\overline{\Diamond}\alpha \equiv_{df} \Diamond\alpha \vee \breve{\Diamond}\alpha$. It is easy to verify that these connectives have the following truth conditions: $\Diamond\alpha$ ($\breve{\Diamond}\alpha$) is true at a world if α holds at some accessible (inaccessible) world; $\overline{\Box}\alpha$ ($\breve{\Diamond}\alpha$) holds iff α holds at all (some) worlds. CO is captured axiomatically as follows.

Definition 1 (Boutilier 1991) The conditional logic CO is the smallest $S \subseteq \mathbf{L}_B$ such that S contains CPL (and its substitution instances) and the following axiom schemata, and is closed under the following rules of inference:

K $\Box(A \supset B) \supset (\Box A \supset \Box B)$
K' $\breve{\Box}(A \supset B) \supset (\breve{\Box}A \supset \breve{\Box}B)$
T $\Box A \supset A$
4 $\Box A \supset \Box\Box A$
S $A \supset \breve{\Box}\Diamond A$
H $\breve{\Diamond}(\Box A \wedge \breve{\Box}B) \supset \breve{\Box}(A \vee B)$
Nes From A infer $\breve{\Box}A$.
MP From $A \supset B$ and A infer B.

Provability and derivability are defined in the standard fashion, in terms of theoremhood.

Theorem 1 (Boutilier 1991) $\vdash_{CO} \alpha$ *iff* $\models_{CO} \alpha$.

Here we consider the extension of CO based on the class of CO-models in which all propositional valuations are represented in W. For all $w \in W$, let w^* be the map from P into $\{0, 1\}$ such that $w^*(A) = 1$ iff $w \in \varphi(A)$; in other words, w^* is the valuation associated with w. Then a CO*-model is a CO-model $M = \langle W, R, \varphi\rangle$ where

$$\{f : f \text{ maps } \mathbf{P} \text{ into } \{0,1\}\} \subseteq \{w^* : w \in W\}$$

CO* has a rather non-standard axiomatization, using a schema adapted from Levesque's (1990) logic OL.

Definition 2 (Boutilier 1991) CO* is the smallest extension of CO closed under all rules of CO and containing the following axioms:

NB $\breve{\Box}\alpha \supset \neg\Box\alpha$ for all falsifiable propositional α.[2]

Theorem 2 (Boutilier 1991) $\vdash_{CO*} \alpha$ *iff* $\models_{CO*} \alpha$.

[1] R is (totally) connected if wRv or vRw for any $v, w \in W$ (this implies reflexivity). This restriction is relaxed in (Boutilier 1992a), where we develop a weaker logic CT4O based on a reflexive, transitive accessibility relation.

[2] Alternatively, we could use $\breve{\Diamond}\alpha$ for all satisfiable α.

2.2 Revision and Subjunctives

An important and well-studied problem in philosophical logic, database theory and artificial intelligence is that of modeling theory change or belief revision. Suppose K is a deductively closed set of beliefs. Revising K is required when new information must be accommodated with these beliefs. If $K \not\models \neg A$, learning A is relatively unproblematic as the new belief set $Cn(K \cup \{A\})$ seems adequate for modeling this change. This process is known as *expansion*. More troublesome is the revision of K by A when $K \models \neg A$. Some beliefs in K must be given up before A can be accommodated. The problem is in determining which part of K to give up, as there are a multitude of choices. Furthermore, in general, there are no logical grounds for choosing which of these alternative revisions is acceptable (Stalnaker 1984), the issue depending largely on context. Fortunately, there are some logical criteria for reducing this set of possibilities.

The main criterion for discarding some revisions in deference to others is that of *minimal change*. Informational economy dictates that as "few" beliefs as possible from K be discarded in order to facilitate belief in A (Gärdenfors 1988). By "few" we intend that, as much as possible, the informational content of K is kept intact. In particular, if KB is a finite representation of K, we do not require that as few sentences as possible from KB be given up, only that the information implicit in these sentences is minimal.[3] While pragmatic considerations will often enter into these deliberations, the main emphasis of the work of Alchourrón, Gärdenfors and Makinson (1985) is in logically delimiting the scope of acceptable revisions. To this end, the AGM postulates below are maintained to hold for any reasonable notion of revision (Gärdenfors 1988). We will use K_A^* to denote revision of K by A, K_A^+ to denote expansion, and $\perp$ to denote the identically false proposition.

(R1) K_A^* is a belief set (i.e. deductively closed).

(R2) $A \in K_A^*$.

(R3) $K_A^* \subseteq K_A^+$.

(R4) If $\neg A \notin K$ then $K_A^+ \subseteq K_A^*$.

(R5) $K_A^* = Cn(\perp)$ iff $\models \neg A$.

(R6) If $\models A \equiv B$ then $K_A^* = K_B^*$.

(R7) $K_{A \wedge B}^* \subseteq (K_A^*)_B^+$.

(R8) If $\neg B \notin K_A^*$ then $(K_A^*)_B^+ \subseteq K_{A \wedge B}^*$.

We can use CO*-models to represent the revision of a propositional theory K. The interpretation of R is as follows: wRv iff v is as *plausible* as w given that K forms our belief set. Plausibility is a pragmatic measure that reflects the degree to which one would accept w as a possible state of affairs given that belief in K may have to be given up. If

[3]Indeed, we do not require that the revised set be equal to the closure of *any* subset of KB (contrast (Nebel 1989) where the syntax, and not the semantic content, of KB is crucial).

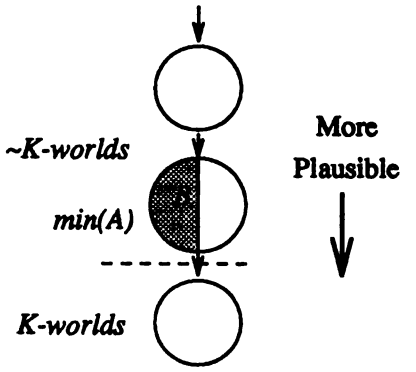


Figure 1: A K-revision model for $A \xrightarrow{KB} B$

v is more plausible than w, loosely speaking, v is "more consistent" with K than w. We can think of each cluster of a CO*-model as the set of worlds having a specified degree of plausibility, with worlds in one cluster being more plausible than those in another just when the second cluster sees the first.

We must insist that those worlds consistent with our belief set K are exactly those minimal in R. That is, vRw for all $v \in W$ iff $M \models_w K$. This condition ensures that *only* worlds consistent with K are maximally plausible, and that *all* K-worlds are equally plausible. We call such models *K-revision models*. The model in Figure 1 is a K-revision model, the bottom cluster consisting of all K-worlds, and those worlds falsifying some element of K distributed among the nonminimal clusters, We refer to these minimal worlds as *(epistemically) possible* and others as *impossible*. Such a constraint can be expressed as

$$\Box(KB \supset (\Box KB \wedge \Box \neg KB))$$

for any K that is finitely expressible as KB. We abbreviate this $O(KB)$ and intend it to mean we "only know" KB (see Section 5). $O(KB)$ partitions a model into two sets of worlds, possible and impossible as described above.

Given this structure, we want the set of minimal A-worlds to represent the state of affairs believed when K is revised by A, since these are the most plausible worlds, the ones we are most willing to adopt, given our acceptance of A. The conditional $A \xrightarrow{KB} B$ is taken to indicate that belief in B results when revision by A occurs, and thus should hold just when B is true at each minimal A-world. This is illustrated in Figure 1. The shaded region is the set of minimal (or most plausible) A-worlds. If B holds at each of these, then $A \xrightarrow{KB} B$ is true of the model. However, even when there are no minimal A-worlds, $A \xrightarrow{KB} B$ should still hold if, at any point on the chain of more plausible worlds, B holds at some A-world w and at all A-worlds more plausible than w. This is captured in the modal language as:

$$A \xrightarrow{KB} B \equiv_{df} \Box \neg A \vee \Diamond(A \wedge \Box(A \supset B)). \quad (1)$$

In this manner, we avoid the *Limit Assumption* (Lewis 1973); there need not exist a *most* plausible A-world for the subjunctive to be meaningful. (Compare Grove's (1988) related semantics for revision where limits are required.)

We define for any $A \in \mathbf{L}_{CPL}$ the belief set resulting from revision of K by A as follows:

$$K_A^{*^M} = \{B \in \mathbf{L}_{CPL} : M \models A \xrightarrow{KB} B\}. \qquad (2)$$

We can show that $*$ satisfies the AGM postulates for belief revision and any AGM revision operator has an equivalent formulation as such a $*$.

Theorem 3 (Boutilier 1992d) *Let M be a K-revision model and $*^M$ the revision function determined by M. Then $*^M$ satisfies postulates (R1) through (R8).*

Theorem 4 (Boutilier 1992d) *Let $*$ be a revision function satisfying postulates (R1) through (R8). Then for any theory K there exists a K-revision model M such that $K_A^* = K_A^{*^M}$ for all A.*

Thus, we can use the logic CO* to represent the revision of a theory KB, and reason about the results of this revision, in a manner respecting the AGM postulates. In fact, CO* appears to be the first purely logical characterization of AGM revision in the sense of providing an explicit language and consequence operator for revision. (Contrast the models discussed in (Gärdenfors 1988) and Grove's (1988) purely semantic account.)

The question of how to revise a theory is important not just in the presence of changing information, but also when we want to investigate questions of the form "What if A were true?" A subjunctive conditional $A > B$ is one of the form[4] "If A were the case then B would be true." Subjunctives have been widely studied in philosophy and it is generally accepted that (some variant of) the *Ramsey test* (Ramsey 1931; Stalnaker 1968) is adequate for evaluating the truth of such conditionals:

> First add the antecedent (hypothetically) to your stock of beliefs; second make whatever adjustments are required to maintain consistency (without modifying the hypothetical belief in the antecedent); finally, consider whether or not the consequent is true. (Stalnaker 1968, p.44)

The connection to belief revision is quite clearly spelled out in this formulation of the Ramsey test: to evaluate a subjunctive conditional $A > B$, we revise our beliefs to include A and see if B is believed. On this view, $A \xrightarrow{KB} B$ is nothing but a subjunctive conditional where the (implicit) KB represents our initial state of knowledge, and will be true exactly when $B \in K_A^*$, in accordance with the Ramsey test.

CO* can be used to represent statements like "If the match were struck it would light" as $M \xrightarrow{KB} L$ and "If the match were struck but wet, it would not light" as $M \wedge W \xrightarrow{KB} \neg L$. The connective is an appropriate representation of such statements, and in (Boutilier 1992d) we develop a framework for subjunctive query answering that improves on existing accounts. In particular, it extends Lewis's (1973) counterfactual logic VC by accounting for factual information in KB reasonably, as a simple example illustrates.

Example Suppose $KB = \{B\}$, a belief set consisting of a single propositional letter. If we were to ask "If A then B?" intuitively we would expect the answer YES, when A is some distinct atomic proposition. With no constraints (such as $A > \neg B$), the postulate of consistent revision (R4) should hold sway and revising by A should result in $KB' = \{A, B\}$. Hence, $A > B$ should be true of KB. Similarly, $\neg(A > C)$ should also be true of KB for any distinct atom C.

In VC there is no mechanism for drawing these types of conclusions. The crucial feature of CO* is once again its ability to represent the fact that KB is all that is known. In this case, given $O(KB)$, both $A \xrightarrow{KB} B$ and $\neg(A \xrightarrow{KB} C)$ are derivable in CO*.

2.3 Normatives and Default Reasoning

We can use CO* for default reasoning as well. In order to define a normative conditional, we impose the following interpretation on the accessibility relation R: world v is accessible to w (wRv) iff v is *at least as normal* as w. Thus, R is an ordering of situations respecting the degree to which an agent judges them to be "normal" or unexceptional. Clusters now contain worlds with identical degrees of normallty rather than plausibility. The truth conditions for $A \Rightarrow B$ can be stated as "In the most normal situations in which A holds, B holds as well:"[5]

$$A \Rightarrow B \equiv_{df} \Box\neg A \vee \Diamond(A \wedge \Box(A \supset B))$$

We write default rules like "birds fly" as $B \Rightarrow F$. Such rules permit exceptions, since, for instance, $B \wedge \neg F$ is consistent with this rule. In (Boutilier 1990; Boutilier 1991) we show that this logic subsumes other conditional approaches to default reasoning. The preferential and rational logics of (Kraus, Lehmann and Magidor 1990; Lehmann 1989) are equivalent to the *simple fragments* of CO and CO*, where we restrict our attention to simple conditionals of the form $A \Rightarrow B$ for $A, B \in \mathbf{L}_{CPL}$. Similarly, the calculus of ε-semantics (Adams 1975; Pearl 1988) is also equivalent to this simple fragment. Though based on probabilistic intuitions, the semantics of this system is remarkably similar to that of CO*. Furthermore, CO* can be used to capture axiomatically the extra-logical solutions proposed for

[4] At least, in "deep structure."

[5] Again, this is only a rough formulation, for it presupposes the *Limit Assumption*, which is not a property required by our definition. There need not be a set of *most* normal worlds satisfying A. The conditional $A \Rightarrow B$ is still meaningful in this circumstance. See (Boutilier 1991; Boutilier 1992d) for details.

the problem of irrelevance, 1-entailment (Pearl 1990) and rational closure (Lehmann 1989).

3 The Ramsey Test and Triviality

The results of the last section show that $\xrightarrow{KB}$ may be interpreted as a subjunctive conditional through appeal to the Ramsey test, relying on a notion of revision respecting the eight AGM postulates. We note an important condition implied by these postulates, *preservation*:

(PR) If $K \not\models A$ then $K^*_A = Cn(K \cup \{A\})$

Thus, if A is consistent with our beliefs, none should be given up when A is learned. This reflects the maxim of informational economy.

While these results seem rather harmless, it soon becomes clear that they stand in apparent contradiction with the Gärdenfors (1986; 1988) *triviality result*. Suppose we allow our belief sets to contain conditionals and insist that these be "believed" in accordance with the Ramsey test:

(RT) $A > B \in K$ iff $B \in K^*_A$.

The triviality result states that (under some natural assumptions) **(PR)** and **(RT)** are incompatible. We describe below a slightly stronger and more concise version of Gärdenfors' result due to Rott (1989).

Let us say a belief set K is AB-*ignorant* if no contingent truth-functional combination of A and B is in K.[6] A revision system is *trivial* if it can sanction no AB-ignorant belief sets, for any propositions A and B. Clearly, trivial revision systems are of little value, for we must allow an agent to abstain from belief in a number of atomic propositions. Unfortunately, once we permit conditional beliefs, all revision systems are trivial if we insist that our revision system satisfy both **(PR)** and **(RT)**; that is, each belief set must contain some belief about A or B. Gärdenfors takes this to indicate that the Ramsey test is not an appropriate acceptability test for conditionals, and this result has attracted much attention (Rott 1989; Grahne 1991; Grahne, Mendelzon and Reiter 1992). It is important to notice at this point that our revision models determine (in general) non-trivial revision functions $*^{\mathbf{M}}$.

In light of triviality, we must examine our results regarding $\xrightarrow{KB}$ and the associated revision functions $*^{\mathbf{M}}$ carefully. The reason CO* does not fall prey to triviality is that our function $*^{\mathbf{M}}$ is defined only on propositional belief sets while the triviality result requires that we allow conditional beliefs. While the revision of K determines belief in conditionals, these are not considered to be part of K itself.

To logically model the elements of a belief set, we will require a belief operator of the type found in standard epistemic logics. Indeed, CO* is a reasonable epistemic logic, as well. We can define a modality for belief $\boxtimes$, reading $\boxtimes\alpha$ as "α is believed." This sentence will hold just when α is true at each epistemically possible world, those minimal in the plausibility ordering R. Hence, we define belief as:

$$\boxtimes\alpha \equiv_{df} \breve{\Diamond}\Box\alpha$$

This means that at some world in the model, α is true at all accessible worlds. This can only be the case when α is true at each world in the minimal cluster (which is accessible to every world).[7] We will have occasion to use this modality here, and we note that it behaves according to the usual weak S5 interpretation of belief (Levesque 1990). For example, the introspection properties are valid:

$$\boxtimes\alpha \supset \boxtimes\boxtimes\alpha \quad \text{and} \quad \neg\boxtimes\alpha \supset \boxtimes\neg\boxtimes\alpha$$

For any CO*-model, we can define the *objective belief set* associated with it to be those propositional sentences that are "believed" in the model.

Definition 3 Let M be a CO*-model. The *objective belief set* associated with M is

$$K = \{\alpha \in \mathbf{L}_{CPL} : M \models \boxtimes\alpha\}$$

We will sometimes refer to this as the propositional belief set or simply the belief set for M. Naturally, the belief set for any K-revision model is just K. We will be more interested in "subjective beliefs" associated with a revision model, those sentences that are believed and involve some modal operators. Of particular concern are those conditionals that are believed by an agent. We therefore extend the notion of belief set to cover arbitrary sentences in $\mathbf{L}_B$.

Definition 4 Let M be a CO*-model. The *extended belief set* associated with M is

$$E = \{\alpha \in \mathbf{L}_B : M \models \boxtimes\alpha\}$$

For any CO*-model with belief set K and extended belief set E, we have $K \subseteq E$.

While extended belief sets may include belief sentences, the truth of a belief sentence is determined solely by the set of epistemically possible worlds and not by the world at which it is being evaluated. This is standard in epistemic logics, and in weak S5 in particular. Since CO* (using $\boxtimes$) captures a weak S5 model of belief, truth of a proposition at the "actual world" has no influence on whether that proposition is believed, for the actual world may not be considered epistemically possible by an agent. In weak S5 (and CO*) $A \wedge \boxtimes\neg A$ is consistent, that is, an agent may have "mistaken beliefs." Similar remarks apply to the conditionals in a belief set. The truth of a conditional is influenced only by the relative ordering of worlds, not by the actual world at which it is being evaluated.

[6] None of $A \vee B$, $\neg A \vee B$, $A \vee \neg B$ or $\neg A \vee \neg B$ are in K

[7] The explicit use of a "belief" modality is also adopted in (Nejdl and Banagl 1992), who also provide an epistemic view of subjunctive queries.

It is desirable to allow extended belief sets containing statements of belief (as in autoepistemic logic below) and conditionals. It is easy to see that once simple (unnested) conditionals are permitted in a belief set we automatically have at least one level of nested belief; for belief in A can be expressed as $\top > A$ for propositional A (Rott 1989). Allowing explicit beliefs of the form $\boxtimes\alpha$ in a belief set is not a novel feature once conditionals are considered. Also notice that if M is a K-revision model generated by KB (i.e., $M \models O(KB)$) then the full introspective ability of our model ensures that $\boxtimes A \in E$ if $KB \models A$ and $\neg \boxtimes A \in E$ otherwise. It is this fact that defeats the intuitive motivation for (PR) once we allow conditionals in our belief sets.

The preservation criterion states that if A is consistent with the elements of a belief set K, then K_A^* should be determined by simply adding A to K (including any further logical consequences). This is quite compelling for propositional belief sets, for it reflects the maxim of informational economy. Preservation, applied to *objective* sets, holds for CO*-models. To see this notice that the minimal A-worlds must be epistemically possible, found in the minimal cluster of worlds, whenever $K \not\models \neg A$ (Boutilier 1992d).

Now, let M be a K-revision model where $K \not\models A$ and $K \not\models \neg A$. We assume that E is the extended belief set for M. It should be clear that $\neg \boxtimes A \in E$, but also that A is *consistent* with E. As is usual in logics of belief, it may be that truth and belief do not correspond. The preservation criterion (PR), if applied to the extended set E, insists that we must simply add A to E and close E under logical consequence. This is clearly inappropriate, for adding A to a belief set during revision means adopting A as a belief (Rott (1989) takes a similar view). But assuming introspective powers of our agent, we should not insist that the belief $\neg \boxtimes A$ be maintained, even though logically A and $\neg \boxtimes A$ are consistent.

There is a conflict between (PR), which incorrectly insists that $\neg \boxtimes A$ be retained, and the natural introspection conditions, which insist that $\neg \boxtimes A$ be rejected in favor of $\boxtimes A$. It is, however, a simple matter to retain the "essential nature" of (PR) while resolving this conflict. We should not label a revision by A a "consistent" revision unless $\boxtimes A$ is consistent with the extended belief set E. Explicitly allowing belief sentences to be part of a belief set implies that we use a different criterion for testing consistency. We can formalize this with the following stronger postulate of preservation for beliefs. Let E now be an *extended* belief set, $A \in \mathbf{L}_{CPL}$, and Cn be (say) consequence in CO*.

(PRB) If $E \not\models \neg \boxtimes A$ then $E_A^* = Cn(E \cup \{A\})$

This is a minimal condition of course, since we ought also include the sentence $\boxtimes A$ explicitly in the revised belief set and so on. However, this definition requires no further exploration since it is vacuous.

Theorem 5 *Let E be an extended belief set. Then $E \not\models \neg \boxtimes A$ iff $E \models \boxtimes A$.*

In other words, if it is consistent to add $\boxtimes A$ to a belief set then it is already present. This discussion can be summarized as follows: *once we allow belief sentences or conditionals in belief sets there can be no "consistent" revisions.*[8]

This demonstrates that the subjunctive $\xrightarrow{KB}$ satisfies the Ramsey test, and that it violates only the "letter of the law" imposed by the postulates. But again, this is only with respect to the "straightforward" extension of the postulates to extended belief sets. Certainly the "spirit" of condition (PR) is retained in our formulation. Indeed, though we no longer have consistent revisions once we make the move to extended belief sets, we still satisfy (PR) on the propositional fragment of these sets. We can define a rather impoverished notion of a *revised* extended belief set E_A^{*M} that extends the definition of K_A^{*M} in a straightforward way. For any E-revision model M:

$$E_A^{*M} = \{B \in \mathbf{L}_B : M \models A \xrightarrow{KB} B\}. \qquad (3)$$

Proposition 6 *Let M be a E-revision model with objective belief set $K \subseteq E$. For any $A \in \mathbf{L}_{CPL}$, if $K \not\vdash A$ then the objective component of E_A^{*M} is $Cn(K \cup \{A\}) = K_A^{*M}$.*

One may notice that since a revision model is suited for a fixed set of beliefs only, the revised extended set E_A^{*M} does not include "new" beliefs or conditionals. Naturally, the true essence of a revised extended set requires that we move to a new revision model M' suitable for K_A^{*M}. Such a system would take us too far afield here, but can be found in (Boutilier 1992c).[9] The semantics of Nejdl and Banagl (1992) provides an account of revised (and *updated*) belief states, and how these may be changed, thus accounting for changing subjective and conditional sentences. This is achieved by providing an ordering relation for each possible world (for update), and each set of possible worlds (for revision). We can think of the revision component of their semantics as a move to a new CO*-model that reflects the new belief set. Their ordering for the new belief set need not bear any relationship to the old ordering, allowing rather arbitrary changes in conditional beliefs. In contrast, the treatment provided in (Boutilier 1992c) preserves as much of this ordering as possible.

While sacrificing (PR) is reasonable in this epistemic setting, it also allows us to keep (RT). It is precisely our strict adherence to the Ramsey test that distinguishes our subjunctive logic from traditional logics, for example, that of Lewis (1973), and allows us to demonstrate close ties to logics for

[8] The undesirable *monotonicity* property entailed by RT (Gärdenfors 1988) asserts that if $K \subseteq L$ then $K_A^* \subseteq L_A^*$. This too is vacuous, since for any extended belief sets, $K \subseteq L$ iff $K = L$.

[9] We are also investigating a system that allows revision of a belief set by *subjective* sentences and conditionals. We have in mind here only *objective* updates. A nice property of this objective update model is that *any* nested conditional (sequence of updates) can be reduced to a simple unnested conditional (single conjunctive update), and this reduction can be computed quite easily).

default reasoning. Before examining this connection, we will explore these differences.

3.1 Comparison to VC

Perhaps the best-known analysis of subjunctives is that of Lewis (1973) (see also Stalnaker (1968)). Our account bears a close relationship to his. Roughly, Lewis maintains that a subjunctive $A > B$ is true if B holds at the A-worlds *most similar* to the actual world. Each world has its own similarity ordering (much like a plausibility ordering), with any world w being more similar to itself than any other world. This semantics leads to *conditional modus ponens*:

CMP $A, A > B \vdash B$.

It is usually claimed that CMP must be valid for the subjunctive conditional. Imagine the subjunctive "If it were raining I would be carrying my umbrella" is true (say, $R > U$). This states that at the most similar worlds to the actual at which R holds, U holds as well. If R is true, clearly the actual world is the most similar world at which R holds. Therefore, for $R > U$ to hold U must be true. Indeed, CMP is valid in most subjunctive logics (e.g., (Lewis 1973; Stalnaker 1968)).

Somewhat at odds with this assessment is the fact that CMP is not valid in CO*:

$$A, A \xrightarrow{KB} B \not\vdash_{CO*} B.$$

In fact, CMP is inappropriate for our "Ramsey style" subjunctives. To be a valid rule of inference, the conclusion of the rule must be true in all situations where the premises hold. This includes both situations considered epistemically possible by an agent and those given little or no plausibility. Formally, this means that for any CO*-model M, *at each world w in this model*, whenever A and $A \xrightarrow{KB} B$ hold, so does B. This is not the case in CO* because of the epistemic nature of the subjunctive conditional. Recall that the truth of $A \xrightarrow{KB} B$ is determined solely by the relative ordering of worlds, not by the particular world w at which it is being evaluated. If B is true at the most plausible A-worlds, then $A \xrightarrow{KB} B$ holds *at each world in the model*. The fact that A and $\neg B$ may be true at w is irrelevant. Indeed, any $A \wedge \neg B$-world in a model M satisfying $A \xrightarrow{KB} B$ (for example, the model in Figure 1) violates the rule CMP.

The key point is that we evaluate a subjunctive by appealing to an explicit reading of the Ramsey test, whereby a subjunctive is believed just when its consequent is present in the state of belief (hypothetically) achieved by obliging the antecedent. In other words, we do not attempt to determine worlds most similar to the *actual world* at which the antecedent holds. Rather we determine worlds that are most plausible given the agent's *actual state of belief*. This is a crucial distinction since an agent's beliefs can potentially bear only a superficial resemblance to the actual world. We detach the truth of $A \xrightarrow{KB} B$ from the truth of A and B precisely as we detach the truth of $\boxtimes A$ from the truth of A.

To return to the umbrella example, given the nature of our connective, $R \xrightarrow{KB} U$ is true in some world just when it is true at all worlds in a CO*-model. This is due to the fact that a CO*-model is suitable for a fixed set of beliefs only. But this means that the truth of $R \xrightarrow{KB} U$ is *independent of the world* at which it is being evaluated, and specifically independent of the truth of R and U at any given world. The conditional is evaluated with respect to a given state of belief, whether these beliefs are true or not. Even if an agent maintains that $R \xrightarrow{KB} U$, it might be the case $R \wedge \neg U$ is true — all that tells us is that the agent's beliefs are mistaken (in particular, it believes $R \supset U$, which is false). Even in worlds where I have forgotten my umbrella on a rainy day $R \xrightarrow{KB} U$ holds *given this particular state of belief*. Imagine being shut up in a windowless office without your umbrella, believing (counterfactually, in your opinion) if it *had* rained today you would have your umbrella, and not realizing that it started raining as soon as you arrived at work. This account of hypothetical deliberation asserts that the truth of a subjunctive (for an agent) is determined completely subjectively. Thus, CMP is inappropriate for this subjunctive conditional.

This is certainly not to suggest that other types of subjunctives cannot be based on or related to the truth of facts in the actual world rather than epistemic states. There must certainly be subjunctives other than those (like $\xrightarrow{KB}$) based on the Ramsey test, conditionals that validate, say, CMP. For example, update semantics proposed for reasoning about revision due to changes in the world instead of changes in belief (Winslett 1990; Katsuno and Mendelzon 1991) might require this rule of inference, and might validate different portions of VC. The logic of Grahne (1991) contains a connective of this sort and bears a strong relationship to VC. Grahne, Mendelzon and Reiter (1992) and Nejdl and Banagl (1992) also provide an update semantics (combined with revision) of this type. But the Ramsey test proffers a different type of conditional, one whose "truth" is subjectively determined relative to particular belief states.

Given the link between the truth conditions for $\xrightarrow{KB}$ and the Ramsey test, this suggests that whenever both $R \xrightarrow{KB} U$ and R are *believed*, U is *believed* as well. This is quite different from CMP, which refers to the *truth* of R and U. Indeed this is valid for CO*:

KMP $\boxtimes A, A \xrightarrow{KB} B \vdash_{CO*} \boxtimes B$

Thus CO* and $\xrightarrow{KB}$ conform more exactly to the intuitions underlying the Ramsey test than do other subjunctive logics, e.g., Lewis's VC. Some expressive advantages of this approach are discussed above and in (Boutilier 1992d).

4 Unifying Normatives and Subjunctives

Aside from providing a view of subjunctives different from the Lewis approach, adopting the Ramsey test allows us to show strong connections between default reasoning and

belief revision. It is somewhat surprising to notice that the definition for the normative conditional $\Rightarrow$ provided in Section 2 is identical to that of $\overset{KB}{\longrightarrow}$. In other words, the subjunctive and normative are formally identical and determine the same inference relation among sets of sentences. A number of people have argued that subjunctives have a natural interpretation as default rules (Ginsberg 1986; Kraus, Lehmann and Magidor 1990; Nejdl 1991; Katsuno and Satoh 1991), but formally identifying the two has been problematic. This is precisely because the classical subjunctive logics satisfy CMP, while logics for normative conditionals typically do not (Kraus, Lehmann and Magidor 1990; Delgrande 1988; Pearl 1990). Indeed, normative conditionals must not verify CMP, for our defaults must allow exceptions. We should be able to say that birds normally fly while a particular bird or class of birds does not.

By adopting the Ramsey test for $\overset{KB}{\longrightarrow}$, we are able to justifiably give up CMP and develop natural normative and subjunctive connectives that have precisely the same formal properties. In this sense, we maintain that the logics governing revision and default reasoning are identical. Assuming that the formal structure of normatives and subjunctives can coincide, the question remains: how do normative and subjunctive reasoning differ? We claim that the distinction lies not in the logic but in the way the logic is applied to reasoning tasks.

4.1 Pragmatic Differences

Consider the evaluation of a normative conditional. To effect this process we must, to some degree, impose an ordering on worlds reflecting the extent to which they are judged to be uniform or unexceptional. The actual world might not be especially homogeneous, but we can imagine what it would be like if it were. In a certain respect, we are aspiring to some idealized norm, some state(s) of affairs that are deemed most normal, at which (for instance) all "defaults" are true and no exceptions exist. It is just these worlds that are minimal in our ordering, and situations that conform more exactly to these uniform states are considered more normal than those that correspond less exactly. Most certainly our knowledge won't allow us to achieve this "epistemic ideal," for inevitably we must give up certain expectations when the real world fails to "cooperate." But to the extent our beliefs are consistent with normality we take advantage, holding onto as many expectations as possible. In this way we are still able to exploit our expectations of the how the world normally (or probably) is, given our knowledge, and make more adventurous predictions. This is precisely the province of default reasoning.

Consider now the evaluation of a subjunctive conditional. Once again we rank states of affairs, but in this case we forgo the ideal used in normative reasoning. We order worlds based on their conformation to the actual state of affairs rather than some unexceptional ideal (or more precisely our *state of belief* about the actual world), relating them to the way things *are* rather than the way things *should be*. We take (what we believe about) the actual world to be the "most normal" or most probable and consider how the world might change given new information.

As normatives constrain expectations at the most normal worlds, so too do subjunctives constrain beliefs at the most plausible (epistemically possible) worlds. A premise $\texttt{bird} \overset{KB}{\longrightarrow} \texttt{fly}$ ensures that $\texttt{bird} \supset \texttt{fly}$ is believed. It must be entailed by KB for any CO*-model of $O(KB)$ verifying the conditional. When we revise our beliefs to include new facts F, we adopt as the new belief state those worlds satisfying F that are minimal in the plausibility ordering. If the new facts are consistent with KB, all beliefs in KB are retained. If not, we are forced to abandon certain beliefs in KB, retaining only those that are verified in the new state of belief. For example, revising by $F = \{\texttt{bird}, \neg\texttt{fly}\}$ requires that the belief $\texttt{bird} \supset \texttt{fly}$ be given up, but revising to include bird alone would ensure that fly is believed.

The analogy to the process of default reasoning should now be clear. When reasoning by default, we attempt to reconcile our information F with the expectations encoded by our default rules. If F is consistent with this set of expectations, each of them is retained; but this is nothing more than revising the set of expectations to include F, since revising a by a consistent sentence is simply adding it to the belief set (in this case, the set of expectations). Similarly, if F is inconsistent, we must relinquish certain expectations. Certain of these will be retained, depending on the ordering of normallty that is represented by our normative conditionals. Again, this is nothing more that revision, for an inconsistent fact forces certain beliefs to be given up, but others are retained according to the ordering of plausibility implicit in our subjunctive premises.

We can formalize this analogy using the normative and subjunctive conditionals in CO*, and show that default reasoning is just the revision of a theory of expectations. Before showing this, we must first illustrate how default priorities and entrenchment of beliefs determine exactly the order in which beliefs or expectations are abandoned.

4.2 Priorities and Entrenchment

Let KB_C be a conditional knowledge base, or a set of default rules $A_i \Rightarrow B_i$. We can view the set of material counterparts $A_i \supset B_i$ of these rules as a *theory of expectations*. We use KB to denote this corresponding set of expectations. If the world were as normal as possible, each of these expectations would be true.[10] Any CO*-model of the rules KB_C satisfies KB at the most normal worlds in that structure. These worlds form our "epistemic ideal." Indeed, given no other knowledge, it seems reasonable to believe each one of these expectations.

Proposition 7 $KB_C \models_{CO^*} \boxtimes KB$

If we are to view default prediction as revision of the theory KB, however, then it should not be the case that KB is merely

[10] Unconditional expectations can be represented as $\top \Rightarrow B_i$.

believed. As discussed in Section 2, the object of revision should be *all* that is believed. Unfortunately, just as a set of subjunctive premises does not determine the belief set being revised, neither does a set normative conditionals logically determine the set of expectations, the set of sentences true at all most normal worlds. This is precisely why the *problem of irrelevance* arises in conditional approaches to default reasoning. A premise $\texttt{bird} \Rightarrow \texttt{fly}$ can be satisfied by a CO*-model with a single most normal world where bird, fly, and $\neg$green each hold. Thus we cannot infer that green birds fly with this as our only premise. While $\texttt{bird} \supset \texttt{fly}$ must be an expectation, the conditional does not rule out additional expectations like $\neg\texttt{green}$.

In (Boutilier 1991) we show that default reasoning based on Pearl's (1990) method of Z-ranking can be captured in CO*. Roughly, default rules, possible worlds, and logical formulae are ranked according to their degree of normality. This ranking (and a corresponding CO*-model) is determined uniquely by a default theory (or set of rules) KB_C. System Z gets around the problem of irrelevance by ensuring that worlds are as normal as possible in the preferred model Z_{KB} (the Z-model for default theory KB_C). We can have no more expectations than are compelled by the conditionals KB_C.

Theorem 8 $Z_{KB} \models O(KB)$

When attempting default prediction, we ask what is true at the most normal situation satisfying some known facts F. In other words, if $F \Rightarrow \alpha$ is derivable, α is a reasonable default conclusion. (We assume F is now propositional.)

If F is consistent with this set of expectations, then *each* expectation is true at the most normal F-worlds. Default prediction in this case is equivalent to simply adding the expectations to F. In other words, $F \Rightarrow \alpha$ iff $F \cup KB \vdash \alpha$. Of course, this is exactly the same as revising KB by F; since $F \cup KB$ is consistent, revision is simply the addition of F to KB, and $F \xrightarrow{KB} \alpha$ just when $F \Rightarrow \alpha$. If F is inconsistent with KB the most normal F-worlds cannot satisfy KB. The epistemic ideal must be relinquished, and certain expectations given up. But once again, we can view this as revision of the theory of expectations. Since $F \cup KB$ is inconsistent, revising KB by F requires that certain beliefs (in this case, expectations) be given up. Again, we will have $F \Rightarrow \alpha$ iff $F \xrightarrow{KB} \alpha$.

Theorem 9 $Z_{KB} \models A \Rightarrow B$ *iff* $Z_{KB} \models A \xrightarrow{KB} B$.

Although seemingly trivial, the key point of this theorem is that the KB being revised implicitly in the sentence $A \xrightarrow{KB} B$ is the theory of expectations KB based on KB_C. This equivalence is only possible once we adopt the Ramsey test for conditionals. But together with Theorem 8, this demonstrates that default reasoning can be viewed as a special case of belief revision, namely, the revision of a theory of expectations to include known facts. Thus we see how the Ramsey test can be applied to default reasoning.

In belief revision deciding which beliefs to keep is determined by an ordering of *epistemic entrenchment* (Gärdenfors 1988). When conflict arises, an agent will give up those beliefs that are less entrenched, keeping those of more importance. A revision function can be specified by means of such an ordering (Gärdenfors 1988). In default reasoning deciding which defaults to violate is determined by means of *priorities*. It turns out that the entrenchment of beliefs can be defined quite naturally in CO* (Boutilier 1992b); and in (Boutilier 1992e) we define priorities for $\Rightarrow$ and show the definition to be correct (but distinct from Pearl's Z-ranking of *rules*). We also show that priorities correspond to the entrenchment of expectations, the key result being summarized as follows:

Theorem 10 (Boutilier 1992e) *The priority ordering on default theory KB_C induced by System Z satisfies the entrenchment postulates of (Gärdenfors 1988). Furthermore, this entrenchment ordering corresponds to the revision function determined by Z_{KB}.*

The entrenchment of beliefs in CO* provides an important generalization of autoepistemic logic (see Section 5).

Another (extra-logical) view of the relationship between default reasoning and belief revision has been put forth by Gärdenfors and Makinson (1990; 1991) and it is compatible with our perspective on the identity of the connectives $\Rightarrow$ and $\xrightarrow{KB}$. They also propose interpreting nonmonotonic inference in terms of revision of one's expectations. Suppose we have some set of defaults KB we are willing to adopt whenever possible. When we wish to determine the nonmonotonic consequences of a sentence A the idea is to revise our expectations KB to include A. The resulting theory is the belief set we are willing to accept given a belief in A. The nature of revision is such that we give up as few defaults as possible from KB to accommodate A. The emphasis of Gärdenfors and Makinson is somewhat different from ours. They take the set of expectations to be primitive and to this add degrees of entrenchment, providing the order in which sentences should be given up. While entrenchment of expectations can be explicitly provided in CO*, we adopt the view that expectations and their associated degrees of entrenchment arise primarily epiphenomenally from an agent's acceptance of default rules or conditionals. From a set of default rules, expectations and entrenchment are automatically derivable.

A similar viewpoint, based on expectations or *hypotheses*, is adopted for default reasoning by Poole (1988). Unfortunately, Poole's Theorist framework does not provide an account of default priorities, thus entrenchment must be specified explicitly. In CO*, entrenchment and priorities can be derived from our defaults.

Some interesting connections are brought to light by the formal equivalence of subjunctive and normative conditionals. Concepts used in one type of reasoning can be expressed and used by the other. For example, the notion of entrenchment and plausibility, commonly used in revision, can be applied

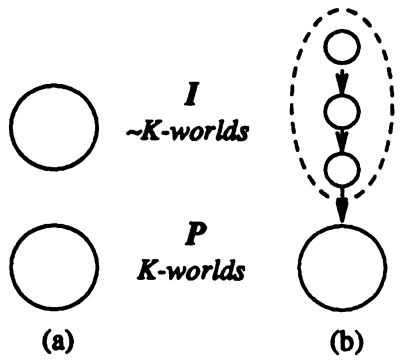


Figure 2: An (a) OL-model and (b) CO*-model

to defaults. In the default domain these concepts correspond to the relative degrees of normality of sentences, or the relative expectation associated with each. Integrity constraints, also used in belief revision (Winslett 1990; Katsuno and Mendelzon 1991; Boutilier 1992d), especially for database update applications, can also be used in default reasoning. In conditional approaches to default reasoning, as described briefly above, irrelevant information has proven problematic (Delgrande 1988; Lehmann 1989; Pearl 1990; Boutilier 1991), yet this has not been identified as a problem in the revision literature. Given a premise $\texttt{bird} \xrightarrow{\text{KB}} \texttt{fly}$, one cannot infer that $\texttt{bird} \wedge \texttt{green} \xrightarrow{\text{KB}} \texttt{fly}$. Updating with irrelevant information can paralyze logical inference, and requires solutions of the type used for default reasoning.

5 Generalized Autoepistemic Logic

Often default statements can be thought of as expressing facts about the consistency of sentences with an agent's current beliefs. Autoepistemic logic (Moore 1985) has been used to model default rules in this way. Levesque (1990) provides a modal logic OL and semantics for autoepistemic logic. An OL-model is determined by a set $\mathcal{P} \subseteq W$ of possible worlds (W being the set of all worlds). $\mathcal{P}$ represents those worlds an agent considers epistemically possible, while the remaining worlds $\mathcal{I} = W - \mathcal{P}$ are epistemically impossible (see Figure 2 (a)). The language $\mathbf{L}_{OL}$ has two modal operators. The sentence $\mathsf{B}\alpha$ is read "α is believed" and is true just when α holds at each possible world. For a fixed structure $\langle \mathcal{P}, w \rangle$ (where $w \in W$ is the actual world), let $\|\alpha\|$ be the set $\{v \in W : \langle \mathcal{P}, v \rangle \models \alpha\}$; that is, the set of worlds satisfying α. Believing α means only α-worlds (those in $\|\alpha\|$) are considered possible. In other words:

$$\langle \mathcal{P}, w \rangle \models \mathsf{B}\alpha \quad \text{iff} \quad \mathcal{P} \subseteq \|\alpha\|.$$

The sentence $\mathsf{O}\alpha$ is read "only α is believed," and represents the fact that an agent believes α, but believes nothing more than α. If an agent knows *more* than α, it must exclude certain α-worlds from its epistemic state. In particular, it must exclude from $\mathcal{P}$ those worlds that make this *additional* knowledge false. If an agent knows nothing more than α, then *all* α-worlds must be considered possible. If some α-world were excluded from $\mathcal{P}$, some additional knowledge would exist corresponding to this exclusion. Thus we say "only α is believed" just when α is believed, and *all* α-worlds are possible:

$$\langle \mathcal{P}, w \rangle \models \mathsf{O}\alpha \quad \text{iff} \quad \mathcal{P} = \|\alpha\|.$$

In fact, Levesque defines $\mathsf{O}\alpha$ in terms of a third connective. $\mathsf{N}\alpha$ is read "at most $\neg\alpha$ is believed" and holds just when α is true at all impossible worlds $W - \mathcal{P}$.

$$\langle \mathcal{P}, w \rangle \models \mathsf{N}\alpha \quad \text{iff} \quad \mathcal{I} \subseteq \|\alpha\|.$$

Given this, $\mathsf{O}\alpha$ can be defined as $\mathsf{B}\alpha \wedge \mathsf{N}\neg\alpha$.

CO*-structures and OL-structures share a great deal in that each consists of a set $\mathcal{P}$ of possible worlds and a set $\mathcal{I}$ of impossible worlds. CO*-models generalize OL-models since the set $\mathcal{I}$ is itself ordered to reflect degrees of impossibility or plausibility (see Figure 2). This relationship enables CO* to duplicate the autoepistemic reasoning of OL. We can define analogues of the connectives N and B in our language. $\mathsf{B}\alpha$ corresponds to α being true at all (possible) worlds in the minimal cluster of M, exactly when $\boxtimes\alpha$ holds. Similarly, $\mathsf{N}\alpha$ corresponds to the truth of α at all (impossible) worlds that are *not* in the minimal cluster. We read $\overleftarrow{\boxtimes}\alpha$ as "At most $\neg\alpha$ is believed," defined as:

$$\overleftarrow{\boxtimes}\alpha \equiv_{df} \boxminus\boxminus\alpha.$$

This simply means that, at every world, α is true at all inaccessible worlds. In particular, since all nonminimal worlds are inaccessible from the minimal worlds, α holds at all nonminimal (impossible) worlds.

We can now translate sentences from $\mathbf{L}_{OL}$ to $\mathbf{L}_{\mathcal{B}}$ in the obvious manner:

Definition 5 Let $\alpha \in \mathbf{L}_{OL}$. The *translation* of α into $\mathbf{L}_{\mathcal{B}}$, denoted α^{Tr}, is defined inductively as

(1) $\alpha^{\text{Tr}} = \alpha$ for atomic proposition α
(2) $(\neg\alpha)^{\text{Tr}} = \neg(\alpha)^{\text{Tr}}$
(3) $(\alpha \supset \beta)^{\text{Tr}} = (\alpha)^{\text{Tr}} \supset (\beta)^{\text{Tr}}$
(4) $(\mathsf{B}\alpha)^{\text{Tr}} = \boxtimes(\alpha)^{\text{Tr}}$
(5) $(\mathsf{N}\alpha)^{\text{Tr}} = \overleftarrow{\boxtimes}(\alpha)^{\text{Tr}}$

If we take $O\alpha$ in CO* to mean $\boxtimes\alpha \wedge \overleftarrow{\boxtimes}\neg\alpha$, we then have:

Theorem 11 *For any* $\alpha, \beta \in \mathbf{L}_{OL}$, $\vdash_{OL} \mathsf{O}\alpha \supset \mathsf{B}\beta$ *iff* $\vdash_{CO*} O(\alpha^{\text{Tr}}) \supset \boxtimes\beta^{\text{Tr}}$.

This demonstrates that autoepistemic reasoning, in the form of queries $\mathsf{O}\alpha \supset \mathsf{B}\beta$ in OL (Levesque 1990), can be faithfully reproduced in CO*. However, the "grades of impossibility" allowed on worlds in CO* determine an entrenchment ordering on beliefs and have implications for the representation of default rules in autoepistemic logic.

A typical default rule such as "birds fly" is represented autoepistemically as $B \wedge \neg\mathsf{B}\neg F \supset F$. Unfortunately, this representation has some undesirable properties. One of these is the fact that $B \wedge E \wedge \neg\mathsf{B}\neg F \supset F$ is derivable, seeming to indicate that emus E also fly by default. Furthermore, if $\neg B$ is believed then the default rule is believed *vacuously*. Thus, $\neg B$ also entails the opposite rule "birds do not fly," $B \wedge \neg\mathsf{B}F \supset \neg F$. These qualities are due to the inability of autoepistemic logic to deal with states of belief other than the actual. "Birds fly" should not be believed just because $\neg B$ is believed. Rather, we ought to consider states of belief that include B when evaluating the default. An autoepistemic reading that accounts for *other* states of belief is the following: "If an agent *were* to believe B, then it *would* believe F." In other words, if an agent revised its beliefs to include the fact B, the resulting belief set would include F. This implies that the belief $B \supset \neg F$ is less entrenched (or more willingly relinquished) than $B \supset F$. (both believed vacuously due to $\neg B$).

This indicates that certain autoepistemic defaults are more naturally viewed as *subjunctive defaults*, for example, $B \xrightarrow{\text{KB}} F$. On this view, the actual state of belief does not have undue influence over the agent's assent to various default rules. This has the added advantage of allowing certain intuitive inferences. New defaults can be derived in a much more reasonable manner. For example, consider the standard "Unemployed Grad Student" example from the default reasoning literature. Reading A, E and S as adult, employed and student respectively, in autoepistemic logic the standard (or straightforward) theory is written as

$$KB = \left\{ \begin{array}{l} A \wedge \neg\mathsf{B}\neg E \supset E \\ S \wedge \neg\mathsf{B}E \supset \neg E \\ S \wedge \neg\mathsf{B}\neg A \supset A \end{array} \right\}$$

Unfortunately, from this theory we can derive defaults stating that student adults are both typically employed and unemployed; both of

$$KB \vdash_{OL} S \wedge A \wedge \neg\mathsf{B}E \supset \neg E$$
$$KB \vdash_{OL} S \wedge A \wedge \neg\mathsf{B}\neg E \supset E$$

hold. In CO*, however, the natural expression of the theory doesn't give rise to this anomaly. Only $S \wedge A \xrightarrow{\text{KB}} \neg E$ is derivable from

$$KB = \left\{ \begin{array}{l} A \xrightarrow{\text{KB}} E \\ S \xrightarrow{\text{KB}} \neg E \\ S \xrightarrow{\text{KB}} A \end{array} \right\}$$

Defaults that are naturally read epistemically can also be represented in CO*. For instance, Etherington's (1988) example about criminal suspects can be written as

$$\texttt{has-motive} \wedge \neg \boxtimes \neg \texttt{guilty} \supset \texttt{suspect}$$

This so-called "default" has nothing to do with revision of beliefs. If someone's guilt is consistent with an agent's beliefs (and the agent believes there is a motive) then they are a suspect. If the agent believes $\neg$`guilty` then `suspect` cannot be inferred. Being a suspect has nothing to do with *potential* states of belief; it is solely a function of the current epistemic state. In this manner we can "mix and match" (traditional) epistemic statements (which are not really default rules at all, as argued by Moore (1985)) with subjunctive defaults. We can show that belief in the subjunctive $A \xrightarrow{\text{KB}} B$ entails (in CO*) belief in the corresponding autoepistemic rule, but not vice versa.

Proposition 12 $A \xrightarrow{\text{KB}} B \vdash_{CO*} \boxtimes(A \wedge \neg \boxtimes \neg B \supset B)$.

Thus, it may be subjunctives that give rise to autoepistemic "rules", which themselves are not sufficient representations of general defaults. The subjunctive nature of the conditional $\xrightarrow{\text{KB}}$ is discussed in (Boutilier 1992d) where we show that epistemic notions are crucial for counterfactual reasoning. Indeed, even subjunctives do not seem to be default rules in the usual sense, as they refer to belief states and not (presumably true) *facts* about the domain in question. Rather they are more like reason-guiding norms adopted *because* of the truth of certain (normative) defaults, and themselves evoke other autoepistemic norms.

6 Concluding Remarks

We have presented a logic in which one can represent normative conditionals, subjunctive conditionals, and autoepistemic statements. This framework has permitted a rigorous comparison of these three types of "default rules" and we have discussed some strong relationships among these. We have shown that the logical behavior of normatives and subjunctives is identical, suggesting a view of default reasoning as revision of a theory of expectations, and that normative and subjunctive defaults logically entail belief in the "corresponding" autoepistemic default, suggesting that autoepistemic beliefs may arise from conditional default rules. These connections were made possible by our strict adherence to the Ramsey test for conditionals, which we claim is tenable in the AGM framework. We simply should not expect (or want) the AGM postulates, as specified, to capture revision of conditional belief sets.

Much work remains to be done to complete the connections discussed here. A reasonable notion of *revised* extended belief sets E^*_A, proposed here simply to illustrate that **(PR)** holds for objective sets in CO*, is currently under investigation (Boutilier 1992c). This provides a reasonable (and non-trivial) semantics for iterated conditionals and revision. The relationship between the orderings of normality and plausibility also needs to be explored. Certainly one's expectations are different from one's beliefs. Beliefs ($\boxtimes\alpha$) must be true at the most plausible worlds, but need not be true at the most normal worlds for an agent. Thus conditionals in a KB must be treated as either subjunctives or normatives, but not both. To reason with both types requires two distinct orderings and is beyond the capabilities of CO*. But these sets and the associated orderings should closely related and a "combined" logic should be feasible.

Acknowledgements: I'd like to thank Gösta Grahne, Alberto Mendelzon, Hector Levesque, Wolfgang Nejdl, Judea Pearl and Ray Reiter for their discussions of this work.

References

Adams, E. W. 1975. *The Logic of Conditionals*. D.Reidel, Dordrecht.

Alchourrón, C., Gärdenfors, P., and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530.

Boutilier, C. 1990. Conditional logics of normality as modal systems. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 594–599, Boston.

Boutilier, C. 1991. Inaccessible worlds and irrelevance: Preliminary report. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 413–418, Sydney.

Boutilier, C. 1992a. Conditional logics for default reasoning and belief revision. Technical Report KRR-TR-92-1, University of Toronto, Toronto. Ph.D. thesis.

Boutilier, C. 1992b. Epistemic entrenchment in autoepistemic logic. *Fundamenta Informaticae*. To appear.

Boutilier, C. 1992c. Iterated conditionals and sequences of updates. Technical report, University of British Columbia, Vancouver. (Forthcoming).

Boutilier, C. 1992d. A logic for revision and subjunctive queries. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 609–615, San Jose.

Boutilier, C. 1992e. What is a default priority? In *Proceedings of Canadian Society for Computational Studies of Intelligence Conference*, pages 140–147, Vancouver.

Delgrande, J. P. 1988. An approach to default reasoning based on a first-order conditional logic: Revised report. *Artificial Intelligence*, 36:63–90.

Etherington, D. W. 1988. *Reasoning with Incomplete Information: Investigations of Non-monotonic Reasoning*. Pitman, London.

Gärdenfors, P. 1986. Belief revisions and the Ramsey test for conditionals. *The Philosophical Review*, 95:81–93.

Gärdenfors, P. 1988. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press, Cambridge.

Gärdenfors, P. and Makinson, D. 1991. Nonmonotonic inference based on expectations. To appear.

Ginsberg, M. L. 1986. Counterfactuals. *Artificial Intelligence*, 30(1):35–79.

Grahne, G. 1991. Updates and counterfactuals. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 269–276, Cambridge.

Grahne, G., Mendelzon, A. O., and Reiter, R. 1992. On the semantics of belief revision systems. In *Proceedings of Theoretical Aspects of Reasoning about Knowledge*, Pacific Grove.

Grove, A. 1988. Two modellings for theory change. *Journal of Philosophical Logic*, 17:157–170.

Katsuno, H. and Mendelzon, A. O. 1991. On the difference between updating a knowledge database and revising it. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 387–394, Cambridge.

Katsuno, H. and Satoh, K. 1991. A unified view of consequence relation, belief revision and conditional logic. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 407–412, Sydney.

Kraus, S., Lehmann, D., and Magidor, M. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207.

Lehmann, D. 1989. What does a conditional knowledge base entail? In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 212–222, Toronto.

Levesque, H. J. 1990. All I know: A study in autoepistemic logic. *Artificial Intelligence*, 42:263–309.

Lewis, D. 1973. *Counterfactuals*. Blackwell, Oxford.

Makinson, D. and Gärdenfors, P. 1990. Relations between the logic of theory change and nonmonotonic logic. In Fuhrmann, A. and Morreau, M., editors, *The Logic of Theory Change*, pages 185–205. Springer-Verlag, Berlin.

Moore, R. C. 1985. Semantical considerations for nonmonotonic logic. *Artificial Intelligence*, 25:75–94.

Nebel, B. 1989. A knowledge level analysis of belief revision. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 301–311, Toronto.

Nejdl, W. 1991. The P-Systems: A systematic classification of logics of nonmonotonicity. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 366–372, Anaheim.

Nejdl, W. and Banagl, M. 1992. Asking about possibilities — revision and update semantics for subjunctive queries. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, Boston. To appear.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo.

Pearl, J. 1990. System Z: A natural ordering of defaults with tractable applications to default reasoning. In Vardi, M., editor, *Proceedings of Theoretical Aspects of Reasoning about Knowledge*, pages 121–135. Morgan Kaufmann, San Mateo.

Poole, D. 1988. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47.

Ramsey, F. P. 1931. *The Foundations of Mathematics and Other Logical Essays*. Kegan Paul, London. (Edited by R. B. Braithwaite).

Rott, H. 1989. Conditionals and theory change: Revisions, expansions, and additions. *Synthese*, 81(1):91–113.

Stalnaker, R. C. 1968. A theory of conditionals. In Harper, W., Stalnaker, R., and Pearce, G., editors, *Ifs*, pages 41–55. D. Reidel, Dordrecht. 1981.

Stalnaker, R. C. 1984. *Inquiry*. MIT Press, Cambridge.

Winslett, M. 1990. *Updating Logical Databases*. Cambridge University Press, Cambridge.

Asking About Possibilities — Revision and Update Semantics for Subjunctive Queries

Wolfgang Nejdl, Markus Banagl
Technische Universität Wien
Christian Doppler Labor for Expert Systems
Paniglgasse 16, A-1040 Vienna, Austria
e-mail: nejdl@vexpert.dbai.tuwien.ac.at

Abstract

The AGM rationality postulates for belief revision play an important role in the formalization of subjunctive queries over knowledge bases. However, the well known triviality results by Gärdenfors in connection with the Ramsey Test make defining a query semantics corresponding to these rationality postulates a difficult task. While existing approaches avoid the triviality results by restricting the postulates in certain ways, we show that an exact translation into a query semantics is possible. The resulting semantics allows for the first time the integration of subjunctive queries evaluated according to a revision semantics corresponding to the AGM rationality postulates with an update operator corresponding to the update semantics formalized by Katsuno, Mendelzon and Grahne. Additionally, the query semantics extends usual conditional implication by defining the concepts of possibility and necessity relative to a knowledge base.

1 Introduction

With the progress in the area of knowledge bases the area of nonstandard query semantics has gained considerable interest. A recent introduction are subjunctive queries ([Demolombe *et al.*, 1991]) which express questions about what the knowledge base deems plausible, were certain facts different from what they are now or more specified. Evaluating such subjunctive sentences using conditional implication ($A \Rightarrow B$) has a long history in various areas starting with [Stalnaker, 1968] and [Lewis, 1973], up to [Gärdenfors, 1988], [Winslett, 1988], [Boutilier, 1990], [Katsuno and Mendelzon, 1991], [Grahne, 1991], [Nejdl, 1991b] and many others.

In many cases, the rationality postulates for belief revision developed by Gärdenfors and colleagues (see for example [Alchourrón *et al.*, 1985], [Makinson, 1985] and [Gärdenfors, 1988]) have been used as a yard stick to compare the different semantics. However, despite apparent similarities between belief revision and the evaluation of sentences containing conditional implication the well-known triviality results (see e.g. [Gärdenfors, 1986] and [Gärdenfors, 1987]) have made it a difficult task to come up with a semantics for conditionals fulfilling all eight AGM postulates. All semantics developed so far restrict the postulates in some way in order to avoid triviality.

However, this is not the only possibility. After discussing the causes for the triviality results in some detail, we show that it is possible to formalize a query semantics agreeing with all rationality postulates. The semantics is based on the concepts of possibility and necessity with respect to the knowledge base and possible revisions and updates thereof. As a side effect, the expressiveness made possible by our semantics allows not only subjunctive queries including conditional implication, but also the concepts of possibility and necessity.

Section 2 shows three examples motivating the need for both revision and update semantics in the sense of [Katsuno and Mendelzon, 1991]. Section 3 discusses the triviality result and its causes in some depths as well as different ways out of this dilemma including short comparisons with existing work. In Section 4 we formalize a query semantics based on the concepts of possibility and necessity relative to belief states and a corresponding conditional implication connective. This semantics allows us to use queries using both update and revision semantics. The update semantics satisfies all Katsuno/Mendelzon postulates from [Katsuno and Mendelzon, 1991] while the revision semantics satisfies all AGM postulates summarized in [Gärdenfors, 1988] whose translations into our semantics are included. Finally, we discuss the relationship of our semantics to five interesting approaches of Grove [Grove, 1988], Katsuno/Satoh [Katsuno and Satoh, 1991], Grahne/Mendelzon/Reiter [Grahne *et al.*, 1992], Morreau [Morreau, 1992] and Boutilier

(who share parts of our motivation but use different ideas and concepts).

2 Introductory Examples

Let us discuss three scenarios in a travel agency. We will consider the travel agent and his database to be one intelligent entity, a view which may become reality in not too far a future, where we seek information from intelligent travel information systems.

Scenario 1 I have told my travel agent, that my vacation starts either on Friday or Saturday and that I want to arrive in Sidney (Australia) on Saturday evening at the latest. After having been told, that this date cannot be guaranteed, I ask the travel agent: "If I were able to depart from Vienna on Friday morning, could my arrival date be guaranteed?" In this case the truth value of the antecedent is not known (as I cannot depart before my vacation starts). If this hypothetical question is answered positive, than the exact start of my vacation does matter and I should check when my vacation really begins. Otherwise this piece of information is worthless. The query therefore determines if it is worthwhile to get some additional piece of information or not.

Scenario 2 In this scenario I have told the travel agent, that my vacation starts on Saturday and have been told, that my arrival date cannot be guaranteed. My question: "If I were able to depart on Friday, would I be able to arrive in Sidney on time?" In this case the antecedent is assumed to be false. If the question is answered positive, I would consider checking if my vacation really started on Saturday as I cannot believe that I overlooked the impossibility of arriving in Sidney on time when planning my vacation. A positive answer therefore determines if I should reconsider the validity of a known piece of information.

Scenario 3 This scenario is quite similar to the second one, except that we both know for certain that my vacation starts on Saturday. The travel agent, sensing that I am to leave his office in despair without having booked a flight, offers me the following bit of advice: "Well, if you could somehow move your vacation to Friday, you would not miss your appointment in Sidney." In this case the antecedent is definitely false, but can be changed by an action such that in the resulting state of affairs I am able to fly on Friday and thus arrive in Sidney on Saturday. This positive answer therefore leads to a set of actions (a plan) that fulfills my goals.

Discussion The three scenarios use different semantics to answer these hypothetical queries. Scenario 1 and 2 use revision semantics, scenario 3 uses update semantics.[1] Revision semantics uses a global ordering of worlds (i.e. relative to a whole belief set) to determine the set of most plausible worlds given some antecedent is true, while update semantics uses an ordering local to each world (model) to determine this set.

In order to give the correct answer in all 3 scenarios, a query semantics both for revision and update is required. As scenario 2 in our example uses only one world to describe the current state of affairs, revision and update semantics coincide. However, unless special constraints are placed on the plausibility ordering for each world, scenario 1 cannot be handled by using update semantics.

3 Triviality Results

3.1 Introduction

Since the reconstruction of Lewis logic for counterfactuals, VC, by Gärdenfors in [Gärdenfors, 1986], the triviality result has played a prominent role in translating rationality postulates on the revision of belief sets like those of [Gärdenfors, 1988] into formalizations compatible to philosophic-linguistic accounts of counterfactual statements (like [Stalnaker, 1968], [Lewis, 1973] and [Nute, 1984]). We will review these results and (extending the arguments from Levi in [Levi, 1988]) discuss the causes leading to triviality and the potential candidates for avoiding triviality.

The task seems simple enough. On the one hand, we have a set of rationality postulates on the revision of belief sets as defined by [Gärdenfors, 1988], on the other hand we have propositional logic extended with an additional connective (conditional implication $\Rightarrow$), where $A \Rightarrow B$ stands for the sentence "If A were true, B would be true". The eight rationality postulate of [Gärdenfors, 1988] are shown below:[2]

$$
\begin{array}{ll}
(K*1) & K_A^* \text{ is a belief set} \\
(K*2) & A \in K_A^* \\
(K*3) & K_A^* \subseteq K_A^+ \\
(K*4) & \neg A \notin K \supset K_A^+ \subseteq K_A^* \\
(K*5) & K_A^* = K_\perp \text{ iff } \models \neg A \\
(K*6) & \models (A \equiv B) \supset K_A^* = K_B^* \\
(K*7) & K_{A \wedge B}^* \subseteq (K_A^*)_B^+ \\
(K*8) & \neg B \notin K_A^* \supset (K_A^*)_B^+ \subseteq K_{A \wedge B}^*
\end{array}
$$

We have now to design a semantics that enables us

[1] The difference between these semantics are formalized in [Katsuno and Mendelzon, 1991] and have been discussed informally independently by [Keller and Winslett, 1985] in the context of relational databases, [Winslett, 1988] in the context of reasoning about action, and [Friedrich *et al.*, 1991] and [Nejdl, 1991a] in the context of model-based diagnosis and repair.

[2] The postulates are explained later in this paper.

to evaluate conditional sentences. Gärdenfors (in [Gärdenfors, 1988]) gives a set of axioms for $\Rightarrow$ following his rationality postulates $K*1$ to $K*8$. The resulting logic is equivalent to VC, the conditional logic formalized by [Lewis, 1973] for evaluating counterfactual sentences. However, as an exact translation of the rationality postulates leads to an inconsistent set of axioms (leading to a series of well known triviality results summarized in [Gärdenfors, 1988]), Gärdenfors replaces $K*4$ by $K*4w$ and $K*8$ by $K*L$ as shown below:

$$(K*4w) \quad A \in K \wedge K \neq K_\perp \supset K \subseteq K^*_A$$
$$(K*L) \quad \neg(A \Rightarrow \neg B) \in K \supset (K^*_A)^+_B \subseteq K^*_{A \wedge B}$$

While this translation shows the strong affinity between belief revision and formalizations of counterfactuals, the question remained as to whether an exact translation would be possible in case certain (implicit) assumptions were dropped.

We will proceed to analyze some important assumptions underlying this triviality result (extending the assumptions listed in [Gärdenfors, 1988] and [Levi, 1988]).

In the following we will use K to represent a knowledge base independent of its actual structure (such as belief sets, belief states etc.). We will make a finer distinction only if necessary.

3.2 Ramsey Test

The translation from belief revision postulates to axioms for conditional implication is usually done using the Ramsey Test formulated in [Ramsey, 1950]. It can be summarized by the following rule:

> (RT) Accept $A \Rightarrow B$ in a belief state K iff the minimal change of K needed to accept A also requires accepting B.

Gärdenfors formalizes RT by

$$(RG) \quad (A \Rightarrow B) \in K \text{ iff } B \in K^*_A$$

and shows that it is incompatible with an exact translation of his rationality postulates.

Another translation of RT is the one by Levi ([Levi, 1988]):

$$(RL) \quad (A \Rightarrow B) \text{ is accepted in } K \text{ iff } B \in K^*_A$$
$$\neg(A \Rightarrow B) \text{ is accepted in } K \text{ iff } B \notin K^*_A$$

The difference to RG is that conditional sentences are not included in K but rather represent judgements of possibility. Further, RL defines the acceptability of negated conditional sentences while RG does not.[3]

[3] If RG did include a similar clause for negated conditionals, even the weakened axioms $K*4w$ and $K*L$ would be too strong and lead to inconsistency.

Excluding conditional sentences from K proves to be a sufficient measure to avoid triviality as discussed already by Levi ([Levi, 1988]). As we do not need conditionals in the knowledge base, this principle will help us to avoid triviality for our query semantics without weakening the rationality postulates. In fact, we can show an exact translation of RL into our semantics, and this work can therefore be viewed as an extension and generalization of Levi's initial suggestions.

3.3 Preservation Criterion

> $(K*P)$ The Preservation Criterion $K*P$ states that propositions B valid in K are also valid in an updated knowledge base K^*_A provided A is consistent with K.

In conjunction with RG it leads to triviality as it is applied to conditional sentences. $K*P$ follows from $K*4$ and is therefore a rather logical criterion to assume. The force of $K*P$ however depends on the contents of K, therefore the contents of the knowledge base plays a prominent role in the avoidance of the triviality result. If we use a Ramsey Test like RL, $K*P$ is only applied to propositional sentences and does not lead to triviality.

3.4 Contents of the Knowledge Base

> (O) K includes sentences containing judgements of possibility and necessity (expressed by $\Rightarrow$, $\Diamond$ or $\Box$).

Although it may seem advantageous to include these sentences in K, they have to be treated with care in order to avoid triviality if we want to achieve an exact translation of the rationality postulates. If we define a semantics for queries over a knowledge base, it seems safe to exclude them from K.

Another possible use for them is to constrain the plausibility ordering of all possible worlds, an option we are currently investigating. In this case they would be treated differently from ordinary facts.

In the context of default reasoning Boutilier ([Boutilier, 1991b]) seems to use them in a similar way. However, as Boutilier treats facts and conditional sentences alike, he either gets wrong results for certain conditionals or has to weaken the rationality postulates (see Section 3.7).

> $(PROP)$ K includes propositional sentences.

This is not valid for some theories of non-monotonic inference operators such as [Kraus *et al.*, 1990]. Of course, defining a query semantics for knowledge bases assumes $PROP$.

3.5 Scope of Conditionals

(*IND*) Acceptability of conditional sentences is defined with respect to a knowledge base containing indeterminate information.

This is the difference between (global) revision semantics and (local) update semantics. Conventional formalizations of conditional logics (like [Lewis, 1973]) as well as query semantics using update semantics (like [Grahne, 1991] and [Grahne and Mendelzon, 1991]) define acceptability of conditional sentences with respect to worlds (equivalent to propositional models, able to express only complete information).

On the other hand, if we want to define a revision semantics corresponding exactly to the AGM rationality postulates the acceptability of conditional sentences and judgements of possibility has to be defined over knowledge bases containing indeterminate information (i.e. over sets of worlds).

3.6 Expressiveness

(NEG) The semantics of negated conditionals has to be specified.

This seems plausible, at least if it does not result in inconsistencies (which is the cause why it has to be excluded from *RG*).

(POS) The query semantics defines concepts of possibility and necessity.

The query semantics should include concepts of possibility and necessity. Indeed, as we show later, conditional implication can be defined using these concepts. This observation has been made already by [Boutilier, 1990], although in the different context of absolute orderings, where we are not able to include these concepts in our query semantics.

(ITER) Conditional sentences can be iterated arbitrarily.

Although this may seem as an advantage at first sight, it is hard to give a plausible semantics to all kinds of iterative sentences. In this paper we do not allow iterations at all, but our semantics can be easily extended to include some plausible kinds of iterations (i.e. $(A \Rightarrow (B \Rightarrow (C \Rightarrow D)))$). Even the opinion that no iteration is necessary is supported by not implausible arguments ([Levi, 1988]).

3.7 Relative vs. Absolute Orderings

(REL) Conditional sentences are evaluated relative to a given knowledge base K.

This is the difference between belief revision and default reasoning. Belief revision semantics is defined with respect to a given knowledge base K which may change in time. As a change of the knowledge base usually implies also a change of the preferred worlds, the plausibility ordering has to depend on the knowledge base and cannot be absolute. This becomes even more clear when we visualize a knowledge base undergoing both revisions and updates.

On the other hand, the idea of default reasoning is that the same set of defaults applies independently from the contents of the knowledge base. Therefore an absolute ordering is sufficient (see also [Nejdl, 1991b]).

This is the reason why the semantics defined by Boutilier in [Boutilier, 1991a] and [Boutilier, 1991b] based on an extension of the modal logic $S4.3$ has interesting applications in default reasoning, but leads to problems when applied to belief revision.

As an example, let $K = (A \vee \neg A) \wedge B$. K consists of the two worlds $\{A, B\}, \{\neg A, B\}$ which are both minimal (and where K corresponds to an $S5$-structure). Additionally, both worlds include $\Diamond A \wedge \Diamond \neg A$. If we revise with A, Boutilier's definition of the conditional $A \Rightarrow B$ simplifies to $\Diamond(A \wedge \Box(A \supset B))$. We therefore include also $A \Rightarrow \Diamond \neg A$ which is certainly not what we expect. Disallowing possibility and necessity in the conditional is a way out of this trouble, but gets us rid of the expressibility of the modal logic as well (an expressiveness we want to keep as much as possible in our approach). Correspondingly, according to Boutilier's comments in his conclusion, he seems to drop $K * P$ and therefore $K * 4$. The resulting logic, however, also does not confirm to Katsuno and Mendelzon's update semantics, so that its exact semantics remains somewhat unclear to us at the moment.

3.8 Structure of the Knowledge Base

(STATE) The structure of the knowledge base supports both global (revision) and local (update) semantics.

In [Gärdenfors, 1988] belief sets are used which include only those propositions whose truth-value is known. Therefore $A \in K$ represents A ***is known to be true***, while $\neg A \in K$ represents A ***is known to be false***. If neither A nor $\neg A$ are in K, the truth value of A is unknown.

A quite different notion is that of a possible world. For every sentence of a given language, either the sentence itself or the negation is true at a certain world i. An exception is the so-called inconsistent world which contains all sentences of the language. Lewis' logic VC ([Lewis, 1973]) is based on possible worlds.

As discussed in [Gärdenfors, 1991], belief sets can be translated to sets of possible worlds and vice versa. However, due to the greater granularity of possible

worlds, some details are lost when translating them into belief sets. In order to support $STATE$, we therefore have to work explicitly with sets of possible worlds (which are basically equivalent to $S5$ modal structures).

We will call such an $S5$ structure representing a knowledge base a *belief state*. Although the states used in [Kraus *et al.*, 1990] have the same structure, they are not suitable for our semantics as their plausibility ordering is not defined on the underlying worlds, but on the states themselves.

In order to support $STATE$, the assumption IND has to be supported as well. This is why systems which define the semantics of conditionals local to each world are unable to express general revision (e.g. [Lewis, 1973], [Grahne, 1991], [Grahne and Mendelzon, 1991]).

4 Query Semantics for Revision and Update

We now define a query semantics conforming to the ideas advocated in Section 3 and discuss the exact translation of the AGM rationality postulates for revision as well as other relevant postulates into (sound) axiom schemas for our semantics. This semantics includes connectives for possibility, necessity and conditional implication relative to belief states as well as an update operator.

4.1 Definitions

Knowledge bases will be represented by *belief states*, which are sets of possible worlds (corresponding to $S5$-structures). Revision semantics will be formalized by using a plausibility ordering on worlds depending on a belief state K ($\leq_K$), update semantics will be formalized by using a plausibility ordering on worlds depending on each world i ($\leq_i$).

Definition 1 (L) L denotes the language of propositional logic.

Definition 2 (Universe W) The universe W is a set of possible worlds i. It includes the inconsistent world $\perp$ where all propositional formulas are true.

These first two definitions are very common, though some approaches do not include the inconsistent world $\perp$.

Definition 3 (Belief States) A belief state K is represented by a set of worlds (i.e. $K \subseteq W$) (corresponding to an $S5$-structure of models of L). The inconsistent world $\perp$ is always a belief state and is called the inconsistent belief state (denoted by $K_\perp$).

Definition 4 ($\leq_K$) A strong plausibility ordering $\leq_K$ for a belief state K is a reflexive, transitive and almost-connected ordering on W.

Defining belief states and an ordering of worlds depending on these belief states is the prerequisite for defining our belief revision semantics. Almost-connectedness is defined as usual as the property that all worlds x, y related by $\leq_K$ are comparable ($x \leq_K y \vee y \leq_K x$).

Definition 5 ($min_{\leq_K}$) Let X an arbitrary subset of W ($X \subseteq W$) and K a belief state. Then $i \in min_{\leq_K} X$ iff

$$\forall j \in (X \setminus min_{\leq_K} X) : (i \leq_K j \wedge j \not\leq_K i) \wedge$$
$$\forall j \in min_{\leq_K} X : (i \leq_K j \wedge j \leq_K i)$$

Definition 6 (Belief revision ordering) Let K be a belief state. A belief revision ordering is a relation $\leq_K$ such that

$$(BRO1) \quad \forall K : (K \neq K_\perp) \supset (K = min_{\leq_K} W)$$
$$(BRO2) \quad \forall K \forall i \in W : (i \neq \perp) \supset (i \leq_K \perp) \wedge (\perp \not\leq_K i)$$

A belief revision ordering satisfies two additional properties, i.e. that the minimal worlds according to $\leq_K$ are those in K and that the inconsistent world is the most implausible one.

Definition 7 ($\|\,\|$) The valuation function $\|\,\|$ assigns to each proposition $A \in L$ a set of worlds $i \in W$, such that $\|A\| = \{i \in W : A \in i\}$.

Remark 1 For an arbitrary $A \in L$, the set $\|A\|$ is always nonempty, because $\perp \in \|A\|$.

The valuation function $\|\,\|$ fulfills the following additional conditions, where $A, B \in L$:

$$(VN1) \quad \|\neg A\| = W \setminus \|A\| \cup \perp$$
$$(VN2) \quad \|A \wedge B\| = \|A\| \cap \|B\|$$

These definitions for the valuation function are the usual ones.

A revision function takes the minimal elements with respect to a belief set K where some new information A is true.

Definition 8 (rev) Let K be a belief state and $A \in L$. Then $rev(K, A) = min_{\leq_K} \|A\|$.

Analogous to *rev* we define an update-function *upd*, expressing the update of information in the knowledge base. For every world $i \in W$, let $\leq_i$ denote a relation that has the same properties as $\leq_K$ (i.e. we have the special case where $K = \{i\}$).

Definition 9 (upd) Let $i \in W$, $A \in L$, and K a belief state. Then $upd(K, A) = \bigcup_{i \in K} min_{\leq_i} \|A\|$.

4.2 Query and Update Types

Using these definitions, we can now define how subjunctive queries are evaluated over a knowledge base KB and how new information is added to it. The ASK and TELL functions we use as primitives for our query language have the following abstract definitions:

- ASK(KB,Query) → yes/no
  where the possible queries will be defined in the next paragraph.
- TELL(KB,A,Type) → KB
  where A is a propositional formula and Type is either "revision" or "update".

The following queries and updates are possible:

Definition 10 (Necessity with respect to K) Let K be a belief state and $A \in L$.

$$\text{ASK(K, } \Box A) = \begin{cases} \text{yes} & \text{if } \forall i \in K : A \in i \\ \text{no} & \text{if } \exists i \in K : \neg A \in i \end{cases}$$

In a more concise notation:

$$\Box_K A \text{ iff } \forall i \in K : A \in i$$

Definition 11 (Possibility with respect to K) Let K be a belief state and $A \in L$.

$$\text{ASK(K, } \Diamond A) = \begin{cases} \text{yes} & \text{if } \exists i \in K : A \in i \\ \text{no} & \text{if } \forall i \in K : \neg A \in i \end{cases}$$

or

$$\Diamond_K A \text{ iff } \exists i \in K : A \in i$$

Definition 12 (Conditional Implication wrt K) Let K be a belief state and $A, B \in L$.

$$\text{ASK(K, } A \Rightarrow_K B) = \begin{cases} \text{yes} & \text{if ASK(rev(K,A), } \Box B) = \text{yes} \\ \text{no} & \text{if ASK(rev(K,A), } \Box B) = \text{no} \end{cases}$$

or

$$A \Rightarrow_K B \text{ iff } \Box_{rev(K,A)} B$$

In the same way we can also define an update conditional with respect to the worlds in K. As this is quite straightforward once we defined everything else, we will not do it in this paper (see also Section 5.2).

Definition 13 (Integrating New Information in K) Let K be a belief state and $A \in L$.

$$\text{TELL(K,A,revision)} = \text{rev(K,A)}$$

or

$$K \circ_R A = rev(K, A)$$

Definition 14 (Changing Information in K) Let K be a belief state and $A \in L$.

$$\text{TELL(K,A,update)} = \text{upd(K,A)}$$

$$K \circ_U A = upd(K, A)$$

The answer to queries containing propositional connectives follows from the definition of the valuation function and corresponds to the usual theorems in propositional logic. For example

$$\text{If (ASK(K,A) = yes) and (ASK(K,B) = yes) then (ASK(K,A} \wedge \text{B) = yes).}$$

This can be rewritten as

$$\Box_K A \wedge \Box_K B \supset \Box_K (A \wedge B)$$

which in fact subsumes all combinations of yes/no answers for the combination $A \wedge B$. Similar axiom schemata can be found for the other connectives.

Besides these axiom schemata which include only queries for the current state of the knowledge base, the second important class of axiom schemata are those which relate queries for the state of the knowledge base after revisions and updates. In Section 5 we will list the valid axiom schemata of this second class. We will use the short form for axioms (using $\Box_K$, $\Diamond_K$ etc.) instead of the long form (using ASK and TELL).

5 Axiom Schemata

Considering our motivation to define a query semantics confirming both the AGM rationality postulates for revision as well as the rationality postulates of [Katsuno and Mendelzon, 1991] and [Grahne, 1991] for update, we will first discuss a translation of the AGM postulates into our semantics and then do the same with the most important update rationality postulates of [Katsuno and Mendelzon, 1991] and [Grahne, 1991].[4] We have therefore the desired result of a query semantics including both revision and update semantics, which is the first such result as far as we know.

5.1 Revision Semantics

5.1.1 Translation Rules

To make the translations clear, let us begin with a brief description of the notation used in [Gärdenfors, 1988].

We have two operations on belief sets, revision and expansion. K_A^* denotes the result of revising the belief set K with sentence A, yielding a new belief set. K_A^+ is defined to be the set $Cn(K \cup \{A\})$, where the function Cn denotes the closure under logical consequences. Hence, to build K_A^+, we simply add A to K

[4] The second part is easier, as our update semantics is defined rather similar to [Grahne, 1991].

and compute all consequences of this set (i.e. expand by A).

$K_\perp$ denotes the inconsistent belief set, i.e. the unique set that contains all wffs of the language L.

A sentence A is in a belief set K iff A is true at every world of the belief state that corresponds to K, i.e. $A \in K$ iff $\Box_K A$. Conversely, $A \notin K$ iff $\neg \Box_K A$.

We model the revision K^*_A by choosing the new belief state to be the set of all A-minimal worlds. Thus, K^*_A translates to $rev(K, A)$, given the definitions above. Finally, $B \in K^+_A$ translates to $\Box_K(A \supset B)$, i.e. $B \in K^+_A$ iff in all worlds of the belief state corresponding to K, B is a logical consequence of A.

5.1.2 Axiom Schemata

In the following we will translate the AGM postulates $K*2$ to $K*8$ into axiom schemata for our query language ($K*1$ is trivial). Additionally, we will translate some additional postulates discussed in Section 3. Note, that K, $rev(K, A)$ and similar expressions denote belief states, whereas A, B, C etc. denote wffs in the propositional language L. The axiom schemata are sound except where otherwise noted. The soundness proofs are included in the longer version of this paper.

K*2 to NB2:

$$(K*2) \quad A \in K^*_A$$

Meaning: If we revise a belief with a sentence A, this sentence is contained in the revised belief set.

Translation: $A \in K^*_A$ iff $A \in rev(K, A)$ iff $\Box_{rev(K,A)} A$ iff

$$(NB2) \quad A \Rightarrow_K A$$

K*3 to NB3:

$$(K*3) \quad K^*_A \subseteq K^+_A$$

Meaning: Revising K with A, the revised belief set contains no sentence that is no consequence of adding A to K.

Translation: $K^*_A \subseteq K^+_A$ iff $B \in K^*_A \supset B \in K^+_A$ iff $\Box_{rev(K,A)} B \supset \Box_K(A \supset B)$ iff

$$(NB3) \quad (A \Rightarrow_K B) \supset \Box_K(A \supset B)$$

K*4 to NB4:

$$(K*4) \quad \neg A \notin K \supset K^+_A \subseteq K^*_A$$

Meaning: If the negation of a sentence A is not in our belief set, that is, we do not know that A is false, then all sentences obtained by adding A to K and performing a closure under consequences are in the revised belief set.

Translation: $\neg A \notin K \supset K^+_A \subseteq K^*_A$ iff $\neg \Box_K \neg A \supset (B \in K^+_A \supset B \in K^*_A)$ iff $\neg \Box_K \neg A \wedge B \in K^+_A \supset B \in K^*_A$ iff

$$(NB4) \quad \neg \Box_K \neg A \wedge \Box_K(A \supset B) \supset (A \Rightarrow_K B)$$

Remark 2 If we exclude $K = K_\perp$, we can substitute $\neg \Box_K \neg A$ by $\Diamond_K A$. However, in the case of $K = K_\perp$, we need the formulation exactly as above, as $\neg \Box_{K_\perp} \neg A$ is not true and we prohibit therefore the derivation of arbitrary conditionals (according to $K*5$).

Remark 3 Note, that NB3 establishes $\Rightarrow_K$ to be weaker than strict implication (defined by modal necessity). NB3 together with NB4 shows it to be equivalent to strict implication in the case where A is possible in K. In the case where A is not possible in K, strict implication does not imply our conditional implication, which helps us to avoid axioms like transitivity, contraposition and strengthening antecedents, which would otherwise follow.

K*5 to NB5:

$$(K*5) \quad \models \neg A \text{ iff } K^*_A = K_\perp$$

Meaning: It is impossible that A is true if and only if revision by A produces an inconsistent belief set.

Translation: $\models \neg A$ iff $\|A\| = \{L\}$ iff $min_{\leq_K} \|A\| = \{L\}$ iff $rev(K, A) = \{L\}$ iff $\Box_{rev(K,A)} \neg A$ iff $A \Rightarrow_K \neg A$.
$K^*_A = K_\perp$ iff $K^*_A \subseteq K_\perp \wedge K_\perp \subseteq K^*_A$ iff (because $K_\perp = L$ and of course $K^*_A \subseteq L$) $K_\perp \subseteq K^*_A$ iff $B \in K_\perp \supset B \in K^*_A$ iff $B \in L \supset B \in K^*_A$ iff (since always $B \in L$) $B \in K^*_A$ iff $A \Rightarrow_K B$.
Hence, K*5 translates to

$$(NB5) \quad (A \Rightarrow_K \neg A) \supset (A \Rightarrow_K B)$$

Remark 4 The other direction — $(A \Rightarrow_K B)$ for all B implies $(A \Rightarrow_K \neg A)$ — is trivially true.

K*6 to NB6:

$$(K*6) \quad \models (A \equiv B) \supset K^*_A = K^*_B$$

Meaning: If A and B are logically equivalent, it does not matter whether we revise with A or B.

Translation: $\models (A \equiv B)$ iff $\|A\| = \|B\|$ iff (because the set $\mathcal{K}$ of all belief states is a superset of W^B, i.e. the set of all singletons $\{i\}$ such that $i \in W$) $\forall K \in \mathcal{K}\ min_{\leq_K} \|A\| = min_{\leq_K} \|B\|$ iff $\Box_{min_{\leq_K} \|A\|} B \wedge \Box_{min_{\leq_K} \|B\|} A$ iff $\Box_{rev(K,A)} B \wedge \Box_{rev(K,B)} A$ iff $(A \Rightarrow_K B) \wedge (B \Rightarrow_K A)$.
$K^*_A = K^*_B$ iff $K^*_A \subseteq K^*_B \wedge K^*_B \subseteq K^*_A$ iff $C \in K^*_A \equiv C \in K^*_B$ iff $\Box_{rev(K,A)} C \equiv \Box_{rev(K,B)} C$ iff $(A \Rightarrow_K C) \equiv (B \Rightarrow_K C)$.
Thus, $K*6$ translates to

$$(NB6) \quad (A \Rightarrow_K B) \wedge (B \Rightarrow_K A) \supset ((A \Rightarrow_K C) \equiv (B \Rightarrow_K C))$$

Remark 5 Another, more direct translation is:

$$(NB6') \quad (\neg(A \equiv B) \Rightarrow_K (A \equiv B)) \supset ((A \Rightarrow_K C) \equiv (B \Rightarrow_K C))$$

K*7 to NB7:

$$(K*7) \quad K^*_{A \wedge B} \subseteq (K^*_A)^+_B$$

Meaning: All sentences obtained by revising with A and B are also obtained by revising with A and then adding B.

Translation: $K^*_{A\wedge B} \subseteq (K^*_A)^+_B$ iff $C \in rev(K, A \wedge B) \supset C \in (rev(K,A))^+_B$ iff $\Box_{rev(K,A\wedge B)}C \supset \Box_{rev(K,A)}(B \supset C)$ iff

$$(NB7) \quad (A \wedge B \Rightarrow_K C) \supset (A \Rightarrow_K (B \supset C))$$

K*8 to NB8:

$$(K*8) \quad \neg B \notin K^*_A \supset (K^*_A)^+_B \subseteq K^*_{A\wedge B}$$

Meaning: Provided that we do not believe that B is false after revising with A, all sentences obtained by revising with A and adding B are also obtained by revising with A and B.

Translation: $\neg B \notin K^*_A \supset (K^*_A)^+_B \subseteq K^*_{A\wedge B}$ iff $\neg\Box_{rev(K,A)}\neg B \supset (C \in (rev(K,A))^+_B \supset C \in rev(K, A \wedge B))$ iff $(A \not\Rightarrow_K \neg B) \supset (\Box_{rev(K,A)}(B \supset C) \supset \Box_{rev(K,A\wedge B)}C)$ iff

$$(NB8) \quad (A \not\Rightarrow_K \neg B) \wedge (A \Rightarrow_K (B \supset C)) \supset (A \wedge B \Rightarrow_K C)$$

K*4w to NB4w:

$$(K*4w) \quad A \in K \wedge K \neq K_\perp \supset K \subseteq K^*_A$$

Meaning: This is the weaker version of $K*4$ used in Gärdenfors reconstruction of VC. It is also used in Grahne's counterfactual logics using update semantics, as $K*4$ degenerates to $K*4w$ if K consists of only one world and therefore includes either A or $\neg A$ (as it is the case when we use the update semantics). We will translate the equivalent version

$$(K*4w')A \in K \supset (K \subseteq K^*_A \vee K = K_\perp)$$

Translation: $A \in K$ iff $\Box_K A$.
$K \subseteq K^*_A$ iff $B \in K \supset B \in K^*_A$ iff $\Box_K B \supset (A \Rightarrow_K B)$.
$K = K_\perp$ iff $\Box_K C$.
Therefore, $K*4w$ translates to

$$(NB4w) \quad \Box_K A \supset ((\Box_K B \supset (A \Rightarrow_K B)) \vee \Box_K C)$$

Remark 6 If we formulate NB4w as

$$(\Box_K A \wedge \Box_K(A \supset B) \supset (A \Rightarrow_K B)) \vee (\Box_K A \supset \Box_K C),$$

we see immediately that $NB4w$ implies $NB4$.

K*P to NBP:

$$(K*P) \quad \neg A \notin K \wedge B \in K \supset B \in K^*_A$$

Meaning: This is the preservation criterion discussed in Section 3. It is implied by $K*4$.

Translation: $\neg A \notin K$ iff $\neg\Box_K\neg A$.
$B \in K$ iff $\Box_K B$.
$B \in K^*_A$ iff $A \Rightarrow_K B$.
Hence, $K*P$ translates to

$$(NBP) \quad \neg\Box_K\neg A \wedge \Box_K B \supset (A \Rightarrow_K B)$$

Remark 7 As $\Box_K B$ implies $\Box_K(A \supset B)$, NBP follows from $NB4$. Like $NB4$, NBP may be instantiated only with propositional sentences A, B, if we want to avoid triviality or meaningless results.

K*M to NBM:

$$(K*M) \quad K \subseteq K' \supset K^*_A \subseteq K'^*_A$$

Meaning: This is the monotonicity criterion which follows from Gärdenfors formalization of the Ramsey Test, RG, and leads to triviality together with $K*P$.

Translation: $K \subseteq K'$ iff $B \in K \supset B \in K'$ iff $\Box_K B \supset \Box_{K'}B$.
$K^*_A \subseteq K'^*_A$ iff $C \in K^*_A \supset C \in K'^*_A$ iff $(A \Rightarrow_K C) \supset (A \Rightarrow_{K'} C)$.
Thus, $K*M$ translates to

$$(NBM) \quad (\Box_K B \supset \Box_{K'}B) \supset ((A \Rightarrow_K C) \supset (A \Rightarrow_{K'} C))$$

This axiom is of course not sound in our semantics (it was the one which lead to the triviality result in the first place.) However a weaker version of it is sound:

$$(NBMw) \quad \Diamond_K A \wedge (\Box_K B \supset \Box_{K'}B) \supset ((A \Rightarrow_K C) \supset (A \Rightarrow_{K'} C))$$

It follows from $NB3$ and $NB4$.

K*L to NBL:

$$(K*L) \quad \neg(A \Rightarrow \neg B) \in K \supset (K^*_A)^+_B \subseteq K^*_{A\wedge B}$$

Meaning: This axiom is the weaker version of $K*8$ which Gärdenfors uses in his axiomatization of VC together with RG. However, $K*8$ and $K*L$ are equivalent if we define the semantics of negated conditionals like in RL.

Translation: $\neg(A \Rightarrow \neg B) \in K$ translates simply to $A \not\Rightarrow_K \neg B$ and must not be translated to $\Box_K(A \not\Rightarrow_K \neg B)$, because formulas containing $\Rightarrow_K$, $\Box_K$, $\Diamond_K$ are not included in K.
Therefore, $K*L$ translates to

$$(NBL) \quad (A \not\Rightarrow_K \neg B) \wedge (A \Rightarrow_K (B \supset C)) \supset (A \wedge B \Rightarrow_K C)$$

which is in fact $NB8$.

5.1.3 Connections to Nonmonotonic Logic

A connection to properties of nonmonotonic inference operators is implied by an interesting paper by Makinson and Gärdenfors ([Makinson, 1991]). They translate the AGM rationality postulates into axioms for nonmonotonic inference relations and get for example the following translation for $K * 4$:

$$(MG4) \quad (T \not\vdash \neg A) \wedge (T \vdash (A \supset B)) \supset (A \vdash C)$$

Using the fact that in our semantics $K = rev(K, \top)$, $\neg \Box_K \neg A \equiv \neg \Box_{rev(K,\top)} \neg A \equiv \neg(\top \Rightarrow_K \neg A)$ and $\Box_K(A \supset B) \equiv \Box_{rev(K,\top)}(A \supset B) \equiv \top \Rightarrow_K (A \supset B)$ the following translation yields exactly our axiom schema $NB4$:

$(\top \not\vdash \neg A) \wedge (\top \vdash (A \supset B)) \supset (A \vdash C)$ iff
$(\top \not\Rightarrow_K \neg A) \wedge (\top \Rightarrow_K (A \supset B)) \supset (A \Rightarrow_K B)$ iff
$\neg \Box_{rev(K,\top)} \neg A \wedge \Box_{rev(K,\top)}(A \supset B) \supset (A \Rightarrow_K B)$ iff

$$(NB4) \quad \neg \Box_K \neg A \wedge \Box_K(A \supset B) \supset (A \Rightarrow_K B)$$

Indeed, all postulates from [Makinson, 1991] can similarly be translated into the corresponding axiom schemata for our query semantics for revision with the exception of $NM5$ and $NM6$. These two seem basically equivalent to our axiom schemata, but cannot be translated using the translation rules described above.

After reflecting upon these similarities between subjunctive conditionals and properties of nonmonotonic inference operators this equivalence should not be too surprising and was already discussed in another context in [Nejdl, 1991b] and [Nejdl, 1992] (though we concentrated on update like conditionals in these papers).

5.2 Update Semantics

As our semantics for updates is based on the same ideas as Grahne's update operator and counterfactual, all axioms listed in [Grahne, 1991] containing only the update operator are valid in our logic. Axioms containing Grahne's counterfactual operators have to be reformulated by using our necessity operator, which is however a rather straightforward task. An update conditional defined similarly to our definition of $\Rightarrow_K$, but using the update function *upd*, would basically lead to the same axioms as Grahne's conditional, i.e. VC^2, although some small differences exist, if we defined it relative to a set of worlds K and not to an individual world i.

In particular, the axioms U1 to U8 for update algebras mentioned in [Grahne and Mendelzon, 1991] (first formalized in [Katsuno and Mendelzon, 1991]) are valid for our query semantics as axiom schemata for update (after appropriate translation, instantiating them only with propositional sentences).

As an example, postulate $U8$

$$(U8) \quad (A \vee B) \circ C \equiv (A \circ C) \vee (B \circ C)$$

translates into

$$NBU8 \quad \Box_{upd(K_1 \cup K_2, A)} B \equiv \Box_{upd(K_1,A) \cup upd(K_2,A)} B$$

or, using our update operator $\circ_U$, into

$$NBU8' \quad (K_1 \vee K_2 \circ_U A) \equiv (K_1 \circ_U A) \vee (K_2 \circ_U A)$$

Especially interesting is the connection of update and revision in our semantics. As connection between update operator and update counterfactuals Grahne uses the following translation of the Ramsey Test:

$$(RR) \quad ((K \circ A) \supset B) \supset (K \supset (A \Rightarrow B))$$

Using our revision operator we get

$$(NBRR_R) \quad ((K \circ_R A) \supset B) \supset (A \Rightarrow_K B)$$

and therefore

$$\Box_{rev(K,A)} B \supset (A \Rightarrow_K B)$$

whose soundness follows immediately from our definition of *rev*.

Using our update operator the translation yields

$$(NBRR_U) \quad \Box_{upd(K,A)} B \supset (A \Rightarrow_K B)$$

which is of course not sound. However, a weaker version is sound

$$(NBRR_U w) \quad \neg \Box_K \neg A \wedge \Box_{upd(K,A)} B \supset (A \Rightarrow_K B)$$

which gives us the connection between update and revision in our query semantics.

6 Additional Related Work

This section is devoted to a short discussion of five other interesting approaches which have been developed independently from the approach discussed in this paper, but share parts of our motivation though using different ideas and concepts. They have been developed by Grove [Grove, 1988], Katsuno/Satoh [Katsuno and Satoh, 1991], Grahne/Mendelzon/Reiter [Grahne *et al.*, 1992], Morreau [Morreau, 1992] and Boutilier [Boutilier, 1992].

The main intention of Grove as described in [Grove, 1988] is to give a semantics for the process of belief revision as defined by Gärdenfors and colleagues, not for subjunctive conditionals. He therefore does not have to consider issues such as the triviality results. His construction is based on a system of spheres that is very similar to the system of Lewis, the main difference being the spheres centered on a set of worlds, rather than on one world. Grove's main result is to show that the AGM-postulates are satisfied by his construction.

A second kind of modeling discussed in his paper based on ordering wffs is basically equivalent.

In contrast, we give a semantics for subjunctive queries guaranteeing a set of theorems for subjunctive conditionals corresponding to the AGM postulates. This allows us to define a query semantics for these conditionals, based on either belief revision or update semantics.

In a similar vain to Grove, Katsuno/Satoh in section 4.2 and 4.3 of [Katsuno and Satoh, 1991] define ordered structures and families of ordered structures and use them to define semantics for revision and update operators, respectively. Again, because they do not consider conditionals they do not have to worry about the Ramsey test.

Morreau (in [Morreau, 1992]) develops a belief revision type semantics for conditionals, so he also faces the triviality results. He claims that the additional assumption "the class of belief sets is closed under expansion" is responsible for them. Belief sets in his paper include both propositions and conditional sentences. Now, as we have expressed with our semantics, the truth value of conditional sentences is uniquely determined by the propositional sentences present in the knowledge base. It follows, that not all combinations of conditional sentences and propositional sentences are possible. So Morreau's assumption can be seen as encoding this consequence of the special relationship of propositional and conditional sentences.

To determine the result of a belief-revision-operation, Morreau uses a technique called "imaging", which is based on an update-ordering (i.e. local order relations). By enforcing the special property "expansiveness", revision semantics is obtained. In contrast to our approach Morreau does not define an operator based on update semantics.

The results of Grahne/Mendelzon/Reiter in [Grahne *et al.*, 1992] are to be seen in the light of a certain philosophical meta-level-concept: Performing a revision by an agent is conceived as update from a viewpoint outside the agent's universe. His revision process changes his knowledge base and is therefore to be modeled as an update-operation on the meta-level. Thus, semantics for update-models have to be considered: Constructing the model-structure, two essential elements are employed. First, local order-relations to define update-semantics in the usual way and second, the Levesque-operators N, B, O, which represent the agent's beliefs. The central result of GMR states that for every belief-revision-system there exists an update-model, such that exactly those sentences χ are logical consequences of $\phi \circ_R \psi$, for which $O\phi \circ_U B\psi \supset B\chi$ is valid in the update-model (where O is the "only-know" operator and B is the "believe" operator). In a sense this is similar to Morreau's construction, which is also based on a local update ordering to define the global revision ordering.

The approach by Boutilier described in [Boutilier, 1992] is based on the modal logic CO^* and defines a belief revision conditional using suitable possibility and necessity operators. Boutilier's approach also avoids the triviality results by restricting his revision models to propositional sentences (also excluding possibility and necessity). However, his operators for possibility and necessity are not indexed by the knowledge base, so that the logic CO^* is only suitable for evaluating conditionals for a fixed knowledge base.

Let us remark finally, that none of these approaches is suitable for the task defined in Section 2 needing conditionals with revision as well as update semantics.

7 Conclusion

We formalized a query semantics which allows for the first time the integration of a revision function according to the rationality postulates of Alchourrón, Gärdenfors and Makinson with an update function according to the rationality postulates of Katsuno and Mendelzon. This query semantics includes concepts of necessity, possibility and conditional implication relative to the knowledge base and the corresponding queries and axiom schemata for revision and update.

We are currently extending these results by writing down a complete set of axiom schemata, by (partially) specifying the plausibility ordering using conditional sentences without including them in the knowledge base and by investigating the complexity of different queries in the average case based on existing worst case results ([Eiter and Gottlob, 1991]). As an additional extension we want to vary the properties of our plausibility ordering according to [Nejdl, 1991b] and investigate the consequences as well as analyze the semantics of iterated conditional sentences in more detail.

An extension to a multi-level model structure which allows us to accommodate both factual knowledge as well as beliefs about how to revise this factual knowledge will be included in the longer version of this paper. Also, a solution to specifying how conditional sentences change after updating or revising the knowledge base is discussed.

References

[Alchourrón *et al.*, 1985] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530, June 1985.

[Boutilier, 1990] Craig Boutilier. Conditional logics of normality as modal systems. In *Proceedings of the National Conference on Artificial Intelligence*

(AAAI), pages 594–599, Boston, August 1990. Morgan Kaufmann Publishers, Inc.

[Boutilier, 1991a] Craig Boutilier. Inaccessible worlds and irrelevance: Preliminary report. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Sidney, August 1991. Morgan Kaufmann Publishers, Inc.

[Boutilier, 1991b] Craig Boutilier. A modal analysis of subjunctive queries. In *Workshop on Nonstandard Queries and Answers*, Toulouse, July 1991.

[Boutilier, 1992] Craig Boutilier. Normative, subjunctive and autoepistemic defaults: Adopting the ramsey test (extended abstract). In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, October 1992. To appear.

[Demolombe *et al.*, 1991]
Robert Demolombe, Luis Fariñas del Cerro, and Tomasz Imielinski, editors. *Workshop on Nonstandard Queries and Answers*, Toulouse, France, July 1991.

[Eiter and Gottlob, 1991] Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactuals. Technical report, Technical University of Vienna, July 1991.

[Friedrich *et al.*, 1991] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. Towards a theory of the repair process. In *Proceedings of the Portuguese Conference on Artificial Intelligence*, Albufeira, October 1991. Springer AI Lecture Notes. Also appeared at the Model-Based Reasoning Workshop, AAAI'91, July 1991, Anaheim.

[Gärdenfors, 1986] Peter Gärdenfors. Belief revisions and the ramsey test for conditionals. *Philosophical Review*, 95:81–93, 1986.

[Gärdenfors, 1987] Peter Gärdenfors. Variations on the ramsey test: More triviality results. *Studia Logica*, 46:319–325, 1987.

[Gärdenfors, 1988] Peter Gärdenfors. *Knowledge in Flux*. MIT Press, 1988.

[Gärdenfors, 1991] Peter Gärdenfors. Belief revision. In *Handbook of Logic in AI and Logic Programming*, chapter 3.2. Oxford University Press, 1991. To appear.

[Grahne and Mendelzon, 1991] Gösta Grahne and Alberto O. Mendelzon. Updates and subjunctive queries. Technical Report KRR-TR-91-4, University of Toronto, 1991.

[Grahne *et al.*, 1992] Gösta Grahne, Alberto O. Mendelzon, and Ray Reiter. On the semantics of belief revision systems. In *Proceedings of the International Conference on Theoretical Aspects of Reasoning about Knowldge*, pages 132–142, Asilomar, March 1992.

[Grahne, 1991] Gösta Grahne. Updates and counterfactuals. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 269–276, Cambridge, MA, April 1991. Morgan Kaufmann Publishers, Inc.

[Grove, 1988] Adam Grove. Two modellings for theory change. *Journal of Philosophical Logic*, 17:157–170, 1988.

[Katsuno and Mendelzon, 1991]
Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, April 1991.

[Katsuno and Satoh, 1991] Hirofumi Katsuno and Ken Satoh. A unified view of consequence relation, belief revision and conditional logic. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 406–412, Sidney, August 1991. Morgan Kaufmann Publishers, Inc.

[Keller and Winslett, 1985] Arthur M. Keller and Marianne Winslett. On the use of an extended relational model to handle changing incomplete information. *IEEE-TSE*, 11(7):620–633, July 1985.

[Kraus *et al.*, 1990] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1–2):167–207, 1990.

[Levi, 1988] Isaac Levi. Iteration of conditionals and the ramsey test. *Synthese*, pages 49–81, 1988.

[Lewis, 1973] D. K. Lewis. *Counterfactuals*. Blackwell, Oxford, 1973.

[Makinson, 1985] D. Makinson. How to give it up: A survey of some formal aspects of the logic of theory change. *Synthese*, 62:347–363, 1985.

[Makinson, 1991] David Makinson. General patterns in nonmonotonic reasoning. In *Handbook of Logic in AI and Logic Programming*, volume 2. Oxford University Press, 1991. To appear.

[Morreau, 1992] Michael Morreau. Epistemic semantics for counterfactuals. *Journal of Philosophical Logic*, 21(1):33–62, 1992.

[Nejdl, 1991a] Wolfgang Nejdl. Belief revision, diagnosis and repair. In *Proceedings of the International GI Conference on Knowledge Based Systems*, München, October 1991. Springer-Verlag.

[Nejdl, 1991b] Wolfgang Nejdl. The P-Systems: A systematic classification of logics of nonmonotonicity. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 366–372, Anaheim, CA, July 1991.

[Nejdl, 1992] Wolfgang Nejdl. The P-Systems: A systematic classification of logics of nonmonotonicity — extended report. Technical report, Technical University of Vienna, January 1992.

[Nute, 1984] Donald Nute. Conditional logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Vol. II*, chapter II.8, pages 387–439. D. Reidel Publishing Company, 1984.

[Ramsey, 1950] F. P. Ramsey. General propositions and casuality. In R. B. Braithwaite, editor, *Foundations of Mathematics and Other Logical Essays*, pages 237–257. Routledge and Kegan Paul, New York, 1950.

[Stalnaker, 1968] Robert Stalnaker. A theory of conditionals. In N. Rescher, editor, *Studies in Logical Theory*. Blackwell, Oxford, 1968. American Philosophical Quarterly Monograph Series, No.2.

[Winslett, 1988] Marianne Winslett. Reasoning about action using a possible models approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 89–93, Saint Paul, Minnesota, August 1988.

Reasoning from Inconsistency: A Taxonomy of Principles for Resolving Conflict

Gadi Pinkas
Dept. of Computer Science
Washington University
Campus Box 1045
St. Louis, MO 63130
pinkas@cics.wustl.edu

Ronald P. Loui
Dept. of Computer Science
Washington University
Campus Box 1045
St. Louis, MO 63130
loui@ai.wustl.edu

Abstract

Systems for non-monotonic reasoning, argument, probabilistic acceptance, and social choice all must define how conclusions arise from conflict. This paper separates the underlying mechanisms that generate conflict and preference from the principle used to resolve conflict. We identify the first-order language in which most principles of conflict resolution can be specified, and we describe the alternatives that result from rearranging quantifiers and connectives. Amazingly, a large fraction of the simple alternatives are reasonable and can be identified in the existing literature. We taxonomize the alternatives by their boldness, and note a duality relation between skeptical and credulous attitudes toward residual conflict.

1 Introduction

Non-monotonic reasoning, systems of argument, probabilistic acceptance rules, and voting rules all must define how conclusions arise from conflict. In each case, the attitude toward the conclusion and the nature of the conflict differs. But similar principles for resolving conflict have occurred in the literatures on each. One prevalent principle is that the conclusion be warranted according to each of the sources, where the sources may be maximal consistent subsets, default theories, or agent preferences. Another principle introduces stratification and seeks agreement among preferred sources. There are other reasonable variations that do not introduce new concepts.

Designers of systems usually choose one way of resolving conflict without much discussion of the pragmatics that led to that choice. The various principles can be ordered according to boldness, and this order is independent of the mechanisms for producing conflict or preference. There are also systems that are permissive toward conflicting possible conclusions and systems that enforce consistency among conclusions. Skepticism and credulity in inheritance systems is one manifestation of this choice.

This paper separates the underlying mechanisms that generate conflict and preference from the principle used to resolve conflict.

A *consequence relation* (CR) is a binary relation that relates a set of beliefs and a conclusion that follows from it. A *consequence specification* is a description (a shorthand) of such a CR. In this paper we are not interested in the details of a specific CR, rather we would like to study the relationships among CR's, their specifications and in particular their relative cautiousness.

Cautiousness is a partial order defined on CR's. A CR is more cautious than another iff the CR's are not equal and every conclusion entailed by the first CR is also entailed by the second one.

The study of cautiousness is important since different degrees of cautiousness may be associated with different styles of decision making. For example, some applications may be forced to make a decision that a certain kind of prudence would not allow. A decision for such an application is critical and there may not be time nor resources to gather more information when there is an ambiguity. In addition, different degrees of cautiousness may result in different degrees of computational complexity. Thus, we may consider using a CR bolder than the ideal, just because it is faster to compute.

Recent literature discusses only two types of cautiousness: skeptical and credulous reasoning [Touretzky et al. 87]. Probabilistic acceptance rules can vary their cautiousness by changing the threshold of minimum acceptable probability; e.g., accept s iff $Prob(s) \geq k$. In this paper, we claim that the space of CR's has an interesting dimension of cautiousness that is not just binary, and is not metrically generated either. Consequence relations may be taxonomized according to their cautiousness.

A language for specifying simple CR's that have oc-

curred in the literature is given, and operators are defined that relate and tie together these classes. The result is a taxonomy of new and existing reasoning paradigms that separates the underlying ideas of conflict and stratification from the many variations for generating conclusions.

2 Consequence relations

Let L be a finite set of well formed formulas (WFFs from either propositional or predicate logic).

A *consequence relation* (CR)[1] is a two-place relation defined over $\mathcal{P}(L) \times L$ (where $\mathcal{P}(L)$ is the power set of L). We use the notation $\Psi R \varphi$ to denote a pair that belongs to the CR R, where Ψ is a finite subset of L and φ is a WFF in L. The subset Ψ is used as a knowledge base while φ is called the conclusion that follows Ψ in R.

A *consequence specification* is a description of a CR in some non-procedural specification language. It is a shorthand used to describe a CR instead of enumerating all the pairs of the relation.

Before proceeding with formal notation, consider simple examples: Given a possibly inconsistent set of WFFs, the CR of classical logic entailment ($\vdash$), trivially entails everything, if indeed there is an inconsistency in the set. However, a more plausible principle may use the following inference procedure: 1) find all maximal consistent subsets; 2) take the closure of each such subset under classical entailment; 3) conclude only the WFFs that are in the intersection of all the closures.

Several systems use this kind of skeptical principle (which we call R_0). This includes some systems dealing with inconsistency [Rescher, Manor 70] and some social choice (voting systems) with the Pareto relation [Campbell 79]. This is how Kyburg [Kyburg61] approaches inconsistency among sentences accepted on high probability. A restricted form of Reiter's nonmonotonic logic [Reiter 80] may also be seen as using the same conflict-resolution principle, if we consider extensions instead of closures of maximal consistent subsets.

Now consider a generalized notion of negation-as-failure on the consequences of R_0; i.e, conclude φ from a set Ψ iff the negation of φ cannot be concluded from Ψ in R_0. If the set of beliefs Ψ is complete[2] (e.g. when using a closed world assumption) then negation-as-failure produces the same set of conclusions as the original CR (R_0). However, if Ψ is incomplete, the result is a very bold CR: it concludes φ from a set Ψ iff there exists a model that satisfies both a maximal consistent subset of Ψ and the WFF φ. We call this CR the dual[3] of R_0 and use the notation $R_0^\dagger$. Derthick's "mundane" reasoning ([Derthick 88]) may be seen as a variation on the dual of R_0.

R_0 is much more cautious than $R_0^\dagger$; any conclusion made by R_0 is also concluded by its dual, and clearly they are not equal. The dual of R_0 is also not *safe* because both φ and $\neg\varphi$ may be concluded from some Ψ. However, if we *clear* the dual from all these ambiguous conclusions we obtain again the original (and safe) R_0.

The duals of safe CR's are good candidates for systems that allow a non-deterministic choice of conclusion. For example, Etherington and Reiter [Etherington] allow any conclusion in any extension, non-deterministically, so long as the result is consistent. Instead of regarding this as a non-deterministic choice of CR, let an unsafe CR represent the non-determinism. Similarly, x may be socially preferred to y if no one prefers y to x (no veto). This principle may be intended to produce a partial order social preference by adopting only one of $x < y$ or $y < x$ when either would be acceptable, by choosing the first option presented. Again, the non-determinism could be in the (unsafe) CR, not a choice of CR's.

Principles that generate unsafe CR's are not only conflict-resolving disputes among prima facie warrants; they also produce conclusions where there was no warrant at all. These CR's can *introduce* conflict instead of resolving it.

3 Operations on CR's and their properties

Formally define the cautiousness relation, duality and the clear operation.

Cautiousness is a partial order defined over CR's:

DEFINITION 3.1 *We say that* $R_1 \succ R_2$ *(R_1 is more* cautious *than R_2) iff* $R_1 \subset R_2$.

If R_1 is more cautious than R_2, then every conclusion available by R_1, is also available by R_2. However, R_2 contains some conclusions that R_1 is too cautious to conclude.

DEFINITION 3.2 *The* dual *of a CR R, written as $R^\dagger$, is the set of of all pairs* $< \Psi, \varphi >$ *such that*

[1] CR's may be defined in a more general and abstract way (like in Scott's information systems [Scott 82]); however for simplicity we prefer to use a restricted by more familiar form.

[2] Ψ is complete in R iff for all φ either $\Psi R \varphi$ or $\Psi R \neg\varphi$.

[3] Since a CR is not a binary relation (its domain is not its co-domain), we are not dismayed that our duality is not the same as duality for graphs. While it is true that relations and objects are not exchanged under this operation, it is still a kind of duality.

$< \Psi, \neg\varphi > \notin R$; *i.e*, $\Psi R^{\dagger} \varphi$ *iff* $\neg(\Psi R \neg\varphi)$.

The dual of a relation corresponds to a generalization of the principle of negation-as-failure. We conclude the negation of a proposition if we can't conclude the proposition itself in the original relation R, and we conclude a proposition in the dual if its negation cannot be concluded in R.

DEFINITION 3.3 *The* clear *operation clears the ambivalent conclusions from the relation; that is, it deletes from R all pairs $\Psi R\varphi$, when both $\Psi R\varphi$ and $\Psi R\neg\varphi$; i.e., $clear(R) = R - \{\Psi R\varphi \mid \Psi R\neg\varphi\}$.*

The dual of the clear of R might be called the *hedge* of R, but we will not discuss this combination further. Figure 1 shows one way of visualizing the clear and dual operations.

We say that R is *safe* if it includes no ambivalent conclusions; i.e., if $\Psi R\varphi$, then $\neg(\Psi R\neg\varphi)$.

The CR's that are generated using the clear and dual operations possess the following properties ([Pinkas, Loui 91]):

1. The dual of a dual is the relation itself; i.e $R^{\dagger\dagger} = R$;
2. The clear of a relation is equal to the clear of the dual of the relation; i.e, $clear(R) = clear(R^{\dagger})$;
3. $clear(clear(R)) = clear(R)$;
4. A conclusion is entailed in $clear(R)$ iff it is entailed by both R and its dual; i.e., $clear(R) = R \cap R^{\dagger}$;
5. The clear of a relation is no less cautious than the relation itself or it dual; i.e, $clear(R) \succeq R$. In fact $clear(R) = R \cap R^{\dagger}$.
6. If R_1 is more cautious than R_2, then the dual of R_2 is more cautious than the dual of R_1; i.e, $R_1 \succ R_2$ iff $R_2^{\dagger} \succ R_1^{\dagger}$.
7. If R is safe then $clear(R) = R$; R is safe and complete iff $R = R^{\dagger}$.

These properties apply to any CR independently of the algorithm or the specification that is used to generate it.

4 A language to specify classes of CR's

In the specification[4] of a CR, we would like to distinguish between the warrant-producing mechanisms that create conflict, and the principle used to determine the CR from those mechanisms (the specification rule). A complete specification of a CR in our language must have both the inference mechanisms and the principle. Our taxonomy is composed of classes where every class is induced by a principle. A CR is mapped into a class if it is possible to identify mechanisms that together with the class principle fully determines the CR. Note that this allows considerable freedom of interpretation. Some CR's can be interpreted as belonging to multiple classes, depending on what is taken to be the mechanism that provides prima facie warrant. To specify principles, consider the language of predicate logic with the standard predicates, functions and axioms for set theory augmented by the following predicates pertaining to:

- Satisfiability: $x \models_{\epsilon} \varphi$ holds iff x is a model that satisfies φ in standard model-theory.
- Theories: $THEORY(\Psi, T)$ is a predicate that asserts that T is a consistent subset of the beliefs in Ψ. For example a theory may be just a plain consistent subset (as in [Rescher, Manor 70], or it may be a subset of beliefs that generates an extension (as in [Reiter 80] or [Touretzky 86]).
- Preference order: $>$ is a transitive, irreflexive, asymmetric order defined on the theories of Ψ. We call T "better" than T' iff $T > T'$. Two theories are equally good or non-comparable if no preference order between them exists. Note that preferences could be probabilistic, refer to social rank, or be purely syntactic, such as specificity among defaults.

The details of the predicate $THEORY$ and the preference order are determined by the mechanism. We assume that the mechanism is fixed; i.e., we know the set of theories and their order for any given Ψ. The particular way to identify theories and their order is not what interests us in this paper; rather, we would like to study the properties of the CR's generated by a variety of principles and independently of the details of the mechanism.

Using the above basic predicates we can define new useful notations:

- Entailment: $\Psi \models \varphi$ holds iff *all* the models that satisfy Ψ also satisfy φ (classical logic entailment).
- Entailment by consistency: $\Psi \Diamond \varphi$ holds iff there exists a model that satisfies both Ψ and φ. i.e, Ψ and φ are consistent (note that $\Diamond$ is the dual of $\models$, since $\neg(\Psi \models \neg\varphi)$ iff $\Psi \Diamond \varphi$).
- Preferred theories (P-theories): $P(\Psi, T)$ holds iff T is a theory of Ψ and there is no theory of Ψ "better"$(>)$ than T.
- The majority quantifier $((\mathcal{M}_C x)Q(x))$ denotes that $Q(x)$ holds for the majority of the x's for

[4] We don't aspire to specify all possible CR's. Instead we would like the language to help us find interesting or useful *classes* of CR's.

which $C(x)$ holds[5]. The dual ($\tilde{\mathcal{M}}$) denotes not-minority; i.e., $(\tilde{\mathcal{M}}_C x)Q(x)$ iff $\neg((\mathcal{M}_C x)\neg Q(x))$.

For example, R_0 is specified by: $\Psi R_0 \varphi$ iff $(\forall T)THEORY(\Psi,T)\rightarrow(T\models\varphi)$, which reads: φ is concluded from Ψ in R_0 iff all the theories of Ψ entail φ. Similarly, $R_0^\dagger$ is specified by: $\Psi R_0^\dagger \varphi$ iff $(\exists T)THEORY(\Psi,T)\wedge(T\Diamond\varphi)$. By using Reiter's extensions as the underlying mechanism, we obtain the skeptical version and the "brave" (credulous) version of Reiter's default logic [McDermott 82].[6] Figure 2 shows several classes of CR's and their relationships. Once a mechanism has been determined, the figure shows the relations among the CR's generated using the different principles. The compact notation in the boxes of figure 2 is itself a shorthand: T represents a theory and P represents a preferred-theory. We therefore use $\forall T\models\varphi$ to denote that all the theories entail φ, and $\exists P\models\varphi$ to denote that at least one preferred theory must entail φ. Figure 2 shows all the CR-classes mentioned in this paper: A solid arrow denotes the cautiousness relation; a dotted arrow denotes the clear operation; the right part of the figure presents the duals of the left part.

Observation: The language is closed under the "clear" and "dual" operations since $\Psi R^\dagger\varphi$ iff $\neg(\Psi R\neg\varphi)$ and $\Psi\ clear(R)\ \varphi$ iff $(\Psi R\varphi)\wedge(\Psi R^\dagger\varphi)$.

5 Other interesting principles

In this section more examples are given for principles. Each box (denote R_i) in figure 2 represents a class of CR's; however, we will treat the R_i's as CR's.

EXAMPLE 5.1 R_2 is a safe CR defined as follows: $\Psi R_2\varphi$ iff all the P-theories (preferred theories) of Ψ entail φ; i.e, $(\forall T)P(\Psi,T)\rightarrow(\mathrm{T}\models\varphi)$. It is a cautious CR, but many researchers find it attractive: [Rescher, Manor 70] suggested indexing methods to determine the preference relation among the maximal consistent subsets. Also, most skeptical nonmonotonic systems may be looked as implementing R_2 (eg. [Touretzky 86]).

The dual of R_2 can be described by: $\Psi R_2^\dagger\varphi$ iff there exists a preferred theory P that is consistent with φ; i.e, $(\exists P)P(\Psi,P)\wedge(P\Diamond\varphi)$. Note that this is the CR that Derthick uses for his "mundane" reasoning. Another example that can easily be mapped is consistency-based diagnosis [Console, Torasso 91], where minimal sets of abnormality assumptions are found that are consistent with the observation. $R_2^\dagger$ is still a very bold CR and of-course $R_2\succ R_2^\dagger$ holds. The clear of both R_2 and its dual is R_2 itself since R_2 is a safe relation. We can also observe that $R_0\succ R_2\succ R_2^\dagger\succ R_0^\dagger$.

EXAMPLE 5.2 The relation R_3 is defined as follows: $\Psi R_3\varphi$ holds iff there exists a preferred theory P of Ψ that entails φ while the rest of the preferred subsets are consistent with φ. Although bolder than R_2, this new CR is still safe and therefore has an advantage over CR's (like the credulous R_4), where a conclusion is justified based on a single preferred theory. The dual of R_3 is: $\Psi R_3^\dagger\varphi$ iff all the preferred theories are consistent with φ or at least one preferred theory entails φ.

EXAMPLE 5.3 $\Psi R_4'\varphi$ holds iff any theory of Ψ entails φ and all better or equally good theories are consistent with φ. Argument systems like Geffner's [Geffner90][7] may be seen as using R_4': An argument (which is a proof based on some consistent subset of beliefs) wins, unless there is a better or equally good ($\geq$) theory that proves the negation. A variation of this CR is described in [Packard, Heiner 83] where a conclusion is made if a consistent subset entails φ and all better or equally good consistent subsets are consistent with φ.

The dual of R_4' is rather weird: $\Psi R_4'^\dagger\varphi$ holds iff either all theories are consistent with φ or if one entails the negation of φ then there exists a better (or equally good) one that entails φ. It is still a safe CR although bolder than R_3. Note that $R_0\succ R_1\succ R_2\succ R_3\succ R_4'\succ R_4'^\dagger\succ R_3^\dagger\succ R_2^\dagger\succ R_1^\dagger\succ R_0^\dagger$.

EXAMPLE 5.4 $\Psi R_4\varphi$ holds iff *a* preferred theory of Ψ entails φ; i.e., $(\exists T)(P(\Psi,T)\wedge(T\models\varphi))$.
It is not a safe CR; however credulous nonmonotonic systems as in [Touretzky et al. 87] correspond to it. The dual of R_4 is: $\Psi R_4^\dagger\varphi$ iff all the preferred theories are consistent with φ. The inference is done by searching for at least one model consistent with φ for any of the preferred theories. Although quite bold, $R_4^\dagger$ is more cautious than Derthick's $R_2^\dagger$. There is no order of cautiousness between R_4 and its dual and since R_4 is not safe, its clear (or the clear of its dual) is R_3.

Some other interesting examples are:

- R_1: There exists a theory that entails φ and all theories are consistent with φ (mentioned in [Packard, Heiner 83], [McDermott 82]).
- R_1': All P-theories entail φ and all the theories are consistent with φ.

[5] Majority is expressed in terms of cardinality: $(\mathcal{M}_C x)Q(x)$ iff $|\{x\mid C(x)\wedge Q(x)\}|>|\{x\mid C(x)\wedge\neg Q(x)\}|$.

[6] Restricted of course because of our too simplistic definitions.

[7] More sophistication is needed from our specification language in order to capture the nuances of a complex argument system like [Simari, Loui 90]. For example we will need recurrence to capture recurrent reinstatement.

- R_2': A P-theory entails φ and all the theories are consistent with φ.
- R_5: A theory entails φ and all better ($>$) theories are consistent with φ.
- R_6: At least one theory entails φ [Reiter 80].
- M: The majority of the theories entail φ. This *safe* CR is useful for social choice systems based on democratic voting (see for example in [Campbell 79]). The dual of M is based on the quantifier "not-minority" ($\tilde{\mathcal{M}}$); i.e, at least half of the theories are consistent with φ.
- M': The majority of the models that satisfy any P-theory also satisfy φ. Although, not even a single theory has to classically entail φ, this CR is *safe*. Its unsafe dual insists that this property holds for at least half of the models (not-minority).

Finally, from a look at figure 2, we can identify three columns: The CR's from the left are called "conservative", since we can find at least one theory that (classically) entails the conclusion. The CR's from the right are called "speculative" since a conclusion is inferred based on the ability to find at least one plausible model that satisfies the conclusion. The third column includes principles that are based on majority (or not-minority) of models.

6 An example: Variations on the maximum entropy mechanism

Due to space limitations we cannot formally map each of the many paradigms that were mentioned in the previous sections. In this section we elaborate on one recent system whose mechanism is not trivially identified. Our purpose is to identify the principle and then to generate a variety of related CR's along the cautious-bold dimension. Doing so reveals a tight relationship with another recent nonmonotonic system.

In [Goldszmidt, et al. 90], given a set of propositional beliefs $\Psi = \{\varphi_i\}$, costs (ρ_i) are computed for each belief, using maximal entropy and epsilon-semantics considerations. Once costs are associated with beliefs, a ranked model [Shoham 88] is constructed in the following way: Let $Vrank_\Psi(\vec{x}) = \sum_{\vec{x} \not\models \rho_i} \varphi_i$; i.e., the rank of a model ($\vec{x}$) is the sum of the costs associated with beliefs, violated by the model. A conclusion φ is entailed from Ψ iff all the models with minimal violation (preferred models) satisfy φ.

It is possible to show that the above principle may be classified as R_2 (as with most nonmonotonic systems): If instead of ranking models, we rank consistent subsets (theories) of beliefs, we obtain a syntactic ranking function on theories: $Trank(\Psi, T) = \sum_{\varphi_i \in \Psi - T} \rho_i$, that is based on summing the costs of beliefs that are not included in the theory. The rank induces a preference order among the theories; i.e., $T < T'$ iff $Trank(T) < Trank(T')$. In [Pinkas 91c], [Pinkas 91h] we proved that φ is entailed using preferred models iff φ is entailed by all the preferred theories. Therefore [Goldszmidt, et al. 90] can be classified as R_2.

Having identified the mechanisms used in [Goldszmidt, et al. 90], a variety of CR's can be generated with the various principles. For example, the credulous version of [Goldszmidt, et al. 90] is based on finding one maximal consistent subset that minimizes the rank and that also entails the conclusion.

The dual of [Goldszmidt, et al. 90] is the system that entails φ iff there exists a preferred theory consistent with φ. Taking again the model-theoretic view, a conclusion is entailed in the dual iff there exist a preferred model that satisfies φ; i.e., there exists a model that minimizes the violation of beliefs and satisfies φ. Surprisingly, this is exactly the mechanism suggested in [Derthick 88]. Therefore, if the costs computed in [Goldszmidt, et al. 90] are taken as the certainties of [Derthick 88], the resulting CR is exactly the CR generated by our generalized notion of negation-as-failure on [Goldszmidt, et al. 90]. In addition, using Levesque's ideas about vivification, if the number of preferred models is reduced to one, then the CR's of [Derthick 88] and [Goldszmidt, et al. 90] collapse and become the same.

Derthick uses a connectionist network to implement his CR. In fact we have shown elsewhere (in [Pinkas, Loui 91], [Pinkas 91g]), that all of the CR's mentioned in this paper (except for the majority-based CR's), can be implemented in a constraint satisfaction network (with soft constraints) which includes Derthick's connectionist architecture as well as other general paradigms like Boltzmann machine and Mean-Field-Theory.

7 Summary

We have shown a taxonomy of consequence relations for reasoning within conflicting beliefs based on their cautiousness. A partial order and several operations defined on CR's reveal an elegant structure that underlies the taxonomy and shed light on the relationships among them. Many existing systems can be mapped into this taxonomy. A simple language demonstrates the separation between the mechanisms for warrant, conflict, and preference and the principles that convert these into a CR. Some interesting new classes (new inference mechanisms) are revealed that deserve further consideration (e.g., $R_1', R_2', R_3, R_4^\dagger, R_6^\dagger, M'$). Some of them are bolder than the pure skeptical mechanism; yet, they are safe and therefore more attractive than credulous systems. Finally, we have considered a case study of a recent nonmonotonic sys-

tem whose inference principle is not trivially identified. Our analysis reveals a tight relationship between two systems that look very different on the surface: [Goldszmidt, et al. 90] and [Derthick 88] use the same inference principle; however, one is the dual of the other and may be generated using negation-as-failure.

System designers, who focus on mechanism and are then faced with the choice of principle, and the inevitable clash of intuitions on principle, can take heart. Depending on the pragmatics of the situation in which the inference system is deployed, any of an ensemble of principles could be specified.

Acknowledgments

We thank Jon Doyle, Fritz Lehmann, and Dave Touretzky, for helpful discussions. The authors were partially supported by NSF grant R-9008012.

References

[Campbell 79] D.E. Campbell, "Manipulation of social choice rules by strategic nomination of candidates," *Theory and Decision 10*, 247-263, 1979.

[Console, Torasso 91] L. Console, P. Torasso, "A spectrum of logical definitions of model-base diagnosis," *Computational Intelligence 7* no, 3, 1991.

[Derthick 88] M. Derthick "Mundane reasoning by parallel constraint satisfaction," PhD thesis, CMU-CS-88-182 Carnegie Mellon University, Sept. 1988

[Etherington, Reiter 84] D. W. Etherington, R. Reiter, "On inheritance hierarchies with exceptions," *Proceedings of The American Associatiation for Artificial Intelligence,* pp. 104-108, Washington D.C. 1984.

[Gabbay 85] D. M. Gabbay, "Theoretical foundations for non-monotonic reasoning in expert systems," In Krzysztof R. Apt, editor, *Proceedings of The NATO Advanced Study Institute on Logics and Models of Concurrent Systems,* pp. 439-457, La Collesur-Loup, France, 1985.

[Goldszmidt, et al. 90] M. Goldszmidt, P. Morris, J. Pearl, "A maximum entropy approach to nonmonotonic reasoning," *Proceedings of AAAI,* pp. 646-652, 1990.

[Tarski] A. Tarski, *Logic, Semantics, Meta-Mathematics. Papers from 1923-1938,* Clarendon Press, Oxford, 1956.

[Touretzky et al. 87] D. Touretzky, R. Thomason, J. Horty, "A clash of intuitions: The current state of NM multiple inheritance systems," Proceedings of the International Conference on Artificial Intelligence, 1987.

[Lehmann 89] D. Lehmann, "What does a conditional knowledge base entail?," *Proc. of The International Conf. on Knowledge Representation and Reasoning,* pp. 212-222, Toronto, Canada, 1989.

[Loui 87] R.P. Loui, "Defeat among arguments: A system of defeasible inference," *Computational Intelligence 3,* no. 3, 1987.

[McDermott 82] D. McDermott, "Non-monotonic logic II: non-monotonic modal theories," *Journal of the ACM 29,* no. 1, 1982.

[McCleland et al. 86] J. L. McCleland, D. E. Rumelhart, G.E Hinton, J.L. McClelland, "The appeal of PDP," in J. L. McClelland and D. E. Rumelhart, *Parallel Distributed Processing: Explorations in The Microstructure of Cognition I,* MIT Press, 1986.

[Packard, Heiner 83] D.J Packard, R.A. Heiner "Inconsistency resolution and collective choise," *Theory and Decision 14,* pp. 225-236, 1982.

[Pinkas 91c] G. Pinkas, "Propositional Non-Monotonic Reasoning and Inconsistency in Symmetric Neural Networks," *Proceedings of IJCAI,* Sydney, 1991.

[Pinkas, Loui 91] G. Pinkas, R. Loui, "Reasoning from inconsistency: A taxonomy and a connectionist approach," technical report WUCS-91-21, Washington University, Computer Science Department, 1991.

[Pinkas 91g] G. Pinkas, "Constructing proofs in symmetric networks," in J. E. Moody, S. J. Hanson, R. P. Lipmann (eds.), to appear in *Advances in Information Processing Systems IV* (NIPS), 1992.

[Pinkas 91h] G. Pinkas, "Representing and learning of symbolic propositional knowledge in symmetric connectionist networks," technical report, Department of Computer Science, Washington University, WUCS-91-52, 1991.

[Rescher, Manor 70] N. Rescher, R. Manor, "On inference from inconsistent premises," *Theory and Decision 1,* pp. 179-217, 1970.

[Rescher 76] N. Rescher, *Plausible Reasoning,* Van Gorcum, 1976.

[Reiter 80] R. Reiter, "A logic for default reasoning," *Artificial Intelligence 13,* pp. 81-132, 1980.

[Scott 82] D.S. Scott, "Domains for denotational semantics," *International Conference on*

Automata, Languages and Programming, 1982.

[Shoham 88] Y. Shoham, *Reasoning about Change*, MIT Press, Cambridge, 1988.

[Simari, Loui 90] G. Simari, R.P. Loui , "Mathematics of defeasible reasoning and its implementation," *AI Journal 52*, no. 2, 1992.

[Touretzky 86] D.S. Touretzky, *The Mathematics of Inheritance Systems*, Pitman, 1986.

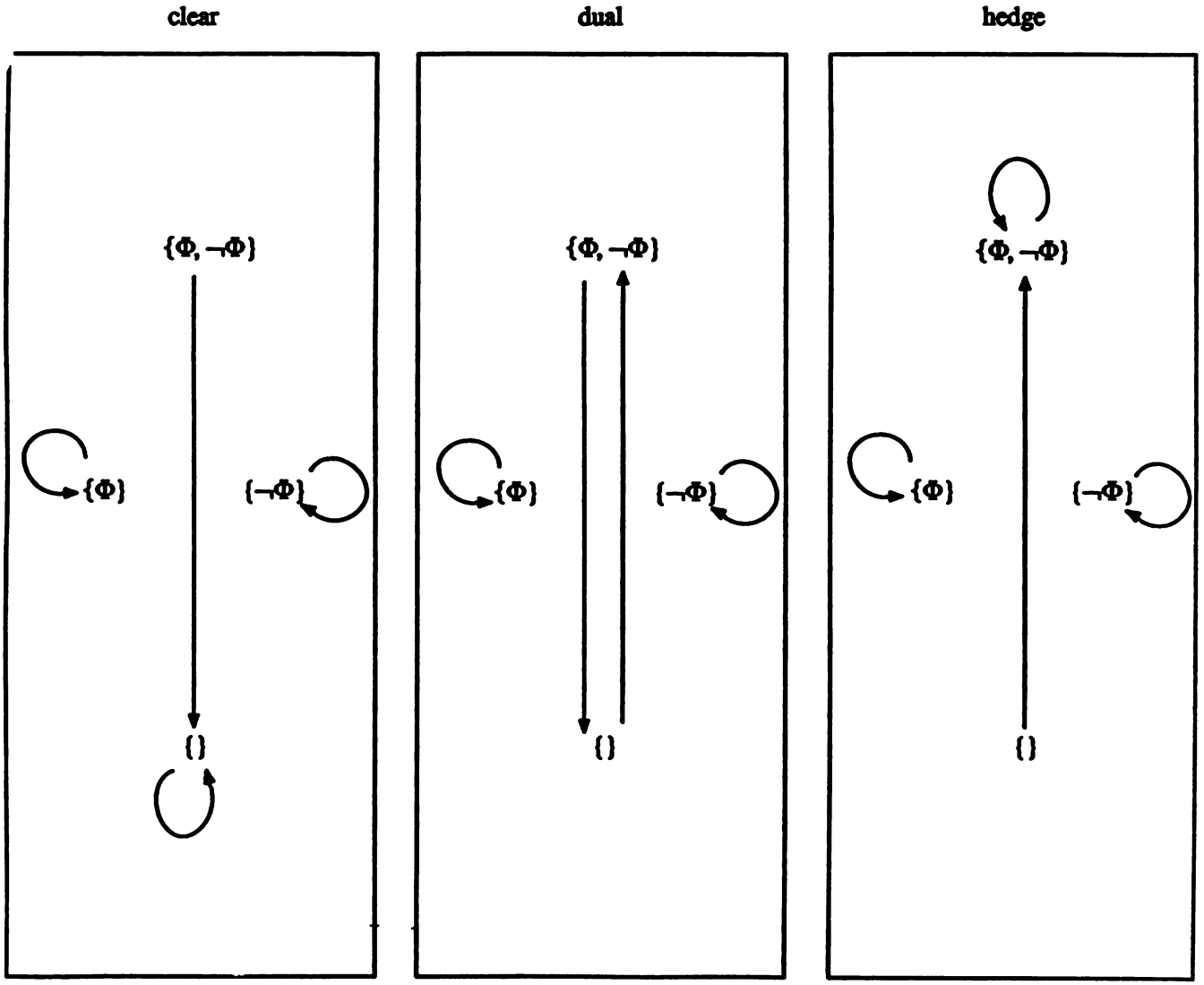


Figure 1. One way of conceptualizing the clear, dual, and hedge (dual of clear) operations. Each subfigure depicts the operation's effect on a single Φ for fixed Ψ. The node label represents (a) that both ΨRΦ and Ψ$R\neg\Phi$, or just one of the two (b, c), or neither (d). The arc shows how the situation is transformed.

Figure 2a. Conservative CRs.
Shaded CRs are unsafe. Their clears are indicated by thin arcs. The cautiousness ordering is indicated by tailed arrows.

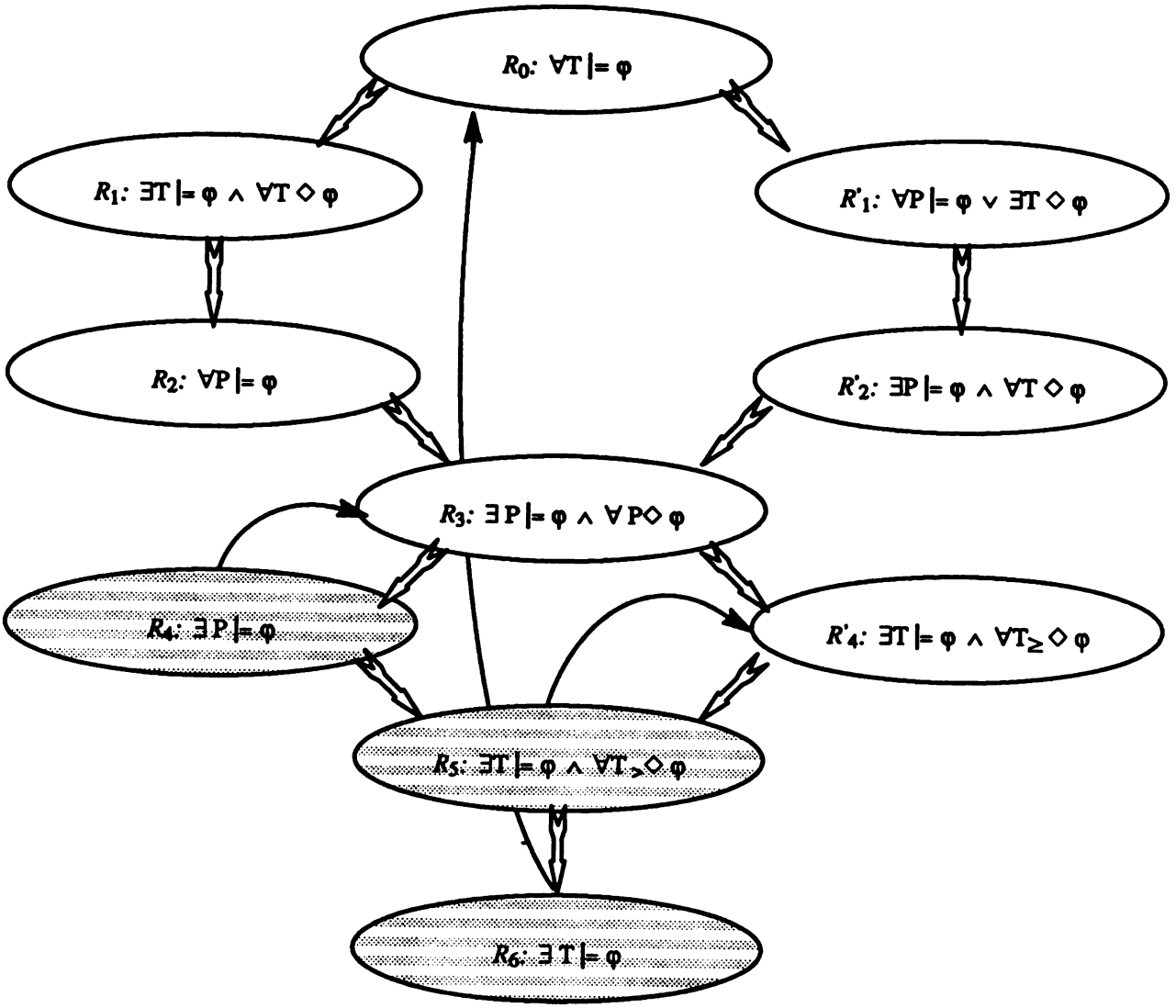


Figure 2b. Speculative CRs.
All are unsafe, hence shaded. Their clears are not shown here. The cautiousness ordering is indicated by tailed arrows.

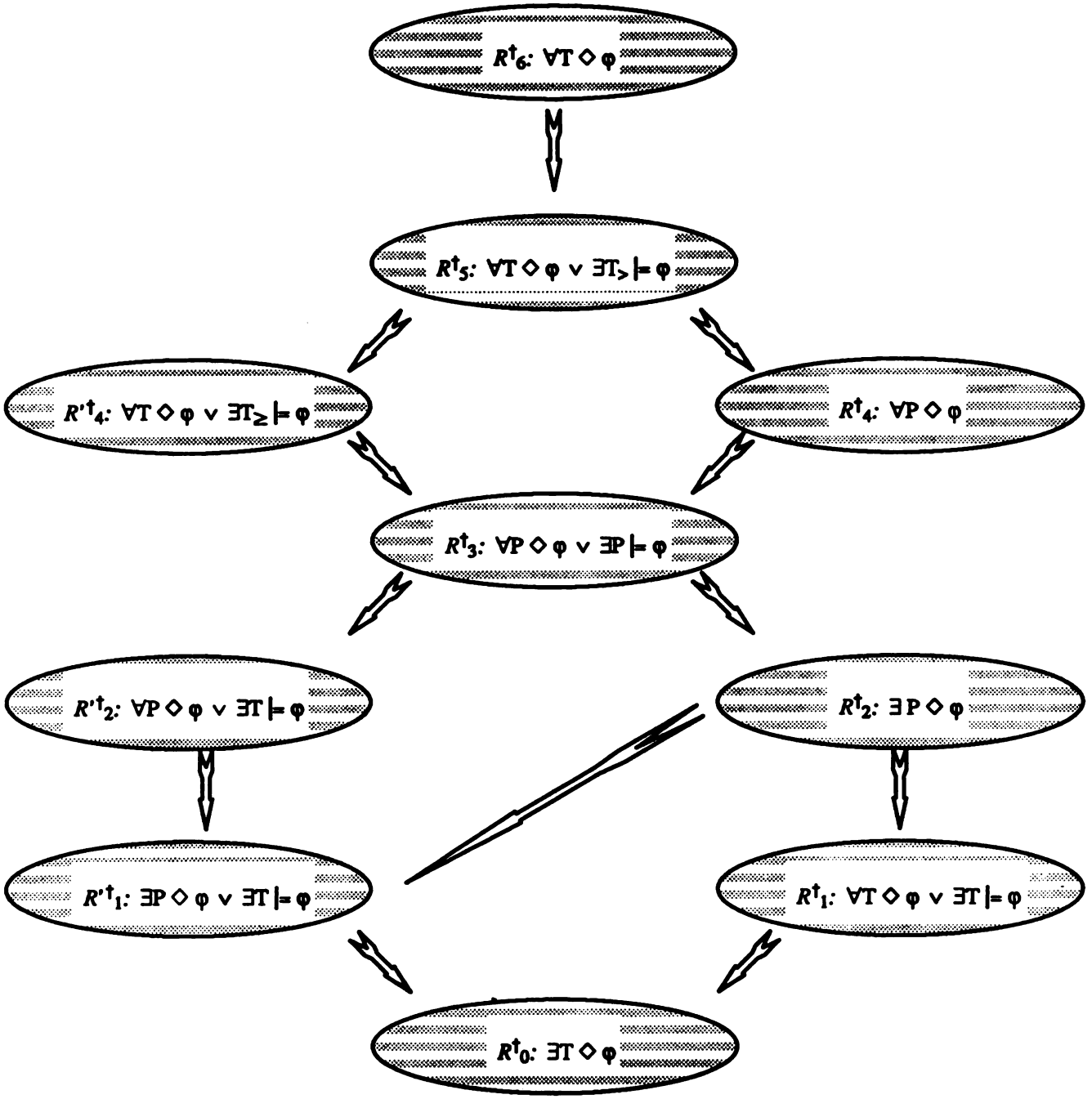
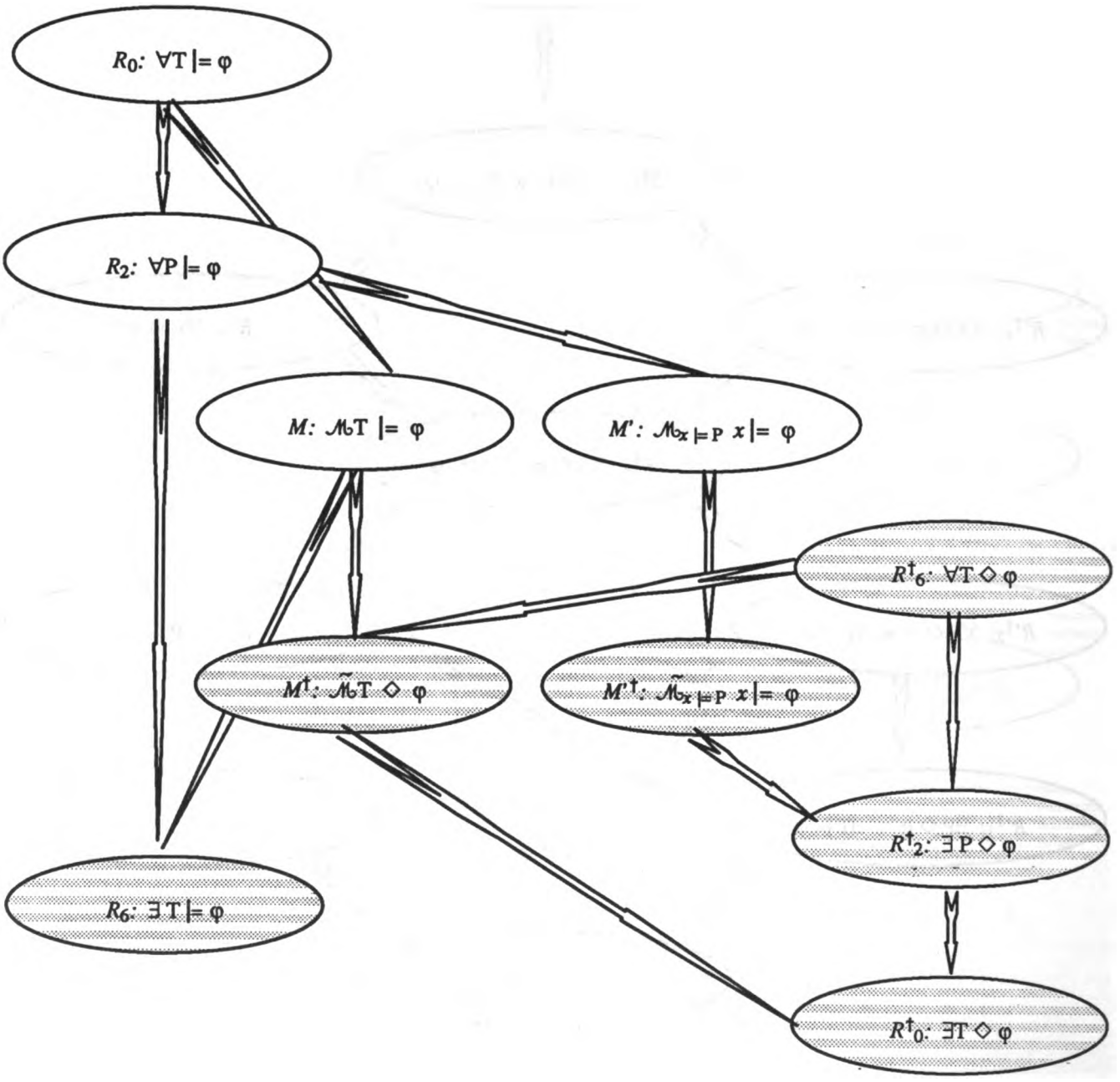


Figure 2c. Majority CRs.



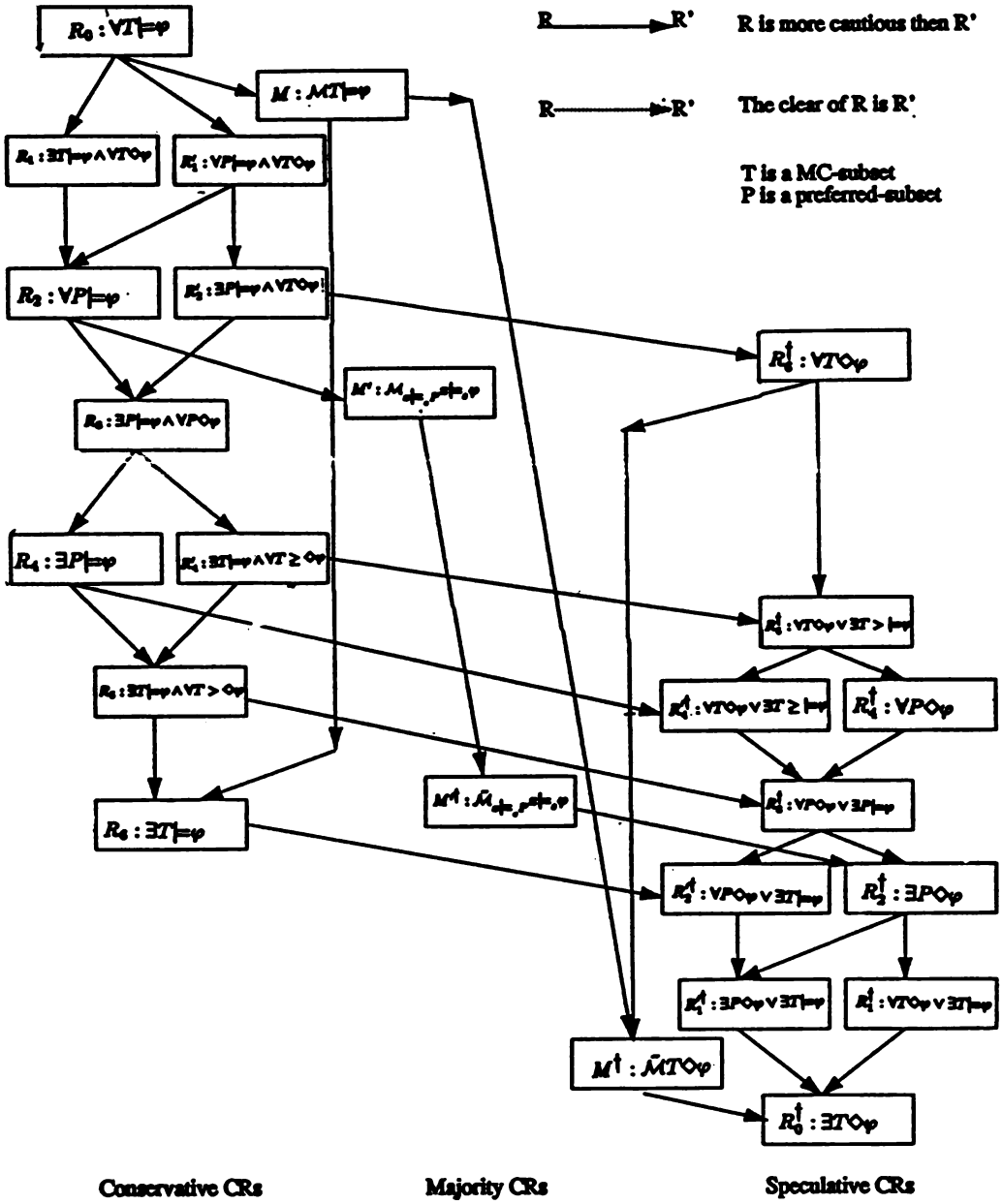


Figure 2d. Figures 2a, 2b, and 2c combined (all links are for one CR more cautious than another; clearing relationships have been omitted).

A Contraction Operator For Classical Propositional Logic

Timothy M. Lownie
Department of Computing and Information Science
Queen's University
Kingston, Canada, K7L 3N6

Abstract

Revising a knowledge-base so that it no longer implies a proposition is called *contraction*. In this paper, a model of contraction is presented based on extending the set of classical truth-values to reflect a flat partial-order. The top elements in the partial-order constitute the classical truth-values; the least element '$\perp$' is interpreted as '*no information*'. Contracting a proposition is then viewed as losing information about the proposition's classical truth. A major contribution of the approach is that semantic distinctions emerge with respect to contractions on inconsistent bases. A comparative analysis with other approaches is presented, including a general analysis covering several topics relevant to KR.

1 INTRODUCTION

This paper considers general semantic problems involved in the operation of revising (propositional) knowledge-bases. The major obstacle in extending updates to closures is resolving ambiguities with respect to contraction operations. To illustrate, consider

$$\alpha \wedge (\alpha \supset \beta) \vdash \beta,$$

where a classical language is assumed. To contract β, either α or $(\alpha \supset \beta)$ must be removed from the left-hand side of '$\vdash$'. However, what are the circumstances by which α is removed before $(\alpha \supset \beta)$? Furthermore, it is not clear that only α or $(\alpha \supset \beta)$ should be excluded in the update. For example, for arbitrary λ,

$$(\beta \vee \lambda) \wedge (\beta \vee \neg\lambda) \vdash \beta.$$

Hence, to contract β, what are the circumstances by which $\beta \vee \lambda$ is removed before $\beta \vee \neg\lambda$, or by which neither is preserved at all? Finally, note that

$$\text{(i) } \alpha \wedge (\alpha \supset \beta) \qquad \text{(ii) } \alpha \wedge \beta$$

are equivalent: the set of consequences connected with either (i) or (ii) is the same. Therefore, (i) and (ii) can be viewed as alternative ways of presenting the same information. Intuitively, contracting β in either case should result in the same update. However, while removing α to obtain an update is intuitive in (i), it is counterintuitive in (ii). Furthermore, identical updates are not guaranteed by removing the same proposition. For example, removing α as a conjunct in (ii) does not even constitute a valid contraction of β.

The objective of this paper is to present a contraction operator for classical propositional logic (CPL). Our proposal is that a satisfactory semantic account is possible by regarding contraction as a *monotone* operator over a partially-ordered set $\{0,1\} \cup \{\perp\}$. The least element in the partial-order ('$\perp$') represents '*no information*'. Informally, the approach interprets the contraction of α by β to mean 'α *with no information about* β'. That is, the contraction is regarded as the information expressed by α under valuations of β to $\perp$. By applying a set of distribution properties, a completeness result can be obtained for the operator, providing an algorithmic basis for computing the classical formula equivalent to a contraction.

In this paper, contraction is viewed as an *informational loss* operation on classical bases, rather than as a collapsing operation on classical theories. Accordingly, the operator's governing role is to be able to selectively update all of the information which is available through query. A principal case involves the situation where information has been introduced into a knowledge-base which renders it *inconsistent*. One of the benefits of the approach is that semantic distinctions are possible with respect to contractions on inconsistent bases. Hence, contractions on distinct inconsistent bases by the same proposition do not have to be equivalent.

2 PREVIOUS WORK

2.1 GÄRDENFORS' POSTULATES

Early investigations by Gärdenfors, Alchourrón, and Makinson (1985 *et al.*) considered two operations: *contraction*, and its dual, *revision*. Their approach has been to propose various intuitive properties ('Gärdenfors' Postulates') in the dynamics of theory change, and to study the class of operators to which these properties correspond. Let '$kb \dot{-} \mu$' refer to the *contraction* of kb by μ; and '$kb\dot{+}\mu$', to the *revision* of kb by μ. Notably, $kb\dot{+}\mu$ can be regarded as an abbreviation for $(kb \dot{-} \neg\mu) \wedge \mu$. That is, to revise kb by μ, kb is contracted by $\neg\mu$ prior to accepting μ. For CPL, Gärdenfors' Postulates for *contraction* may be summarized as follows (Katsuno, Mendelzon 1990b):

(C1) $kb \vdash kb \dot{-} \mu$ (*inclusion*)

(C2) if $kb \not\vdash \mu$ then $kb \dot{-} \mu \equiv kb$ (*vacuity*)

(C3) if $\not\vdash \mu$ then $kb \dot{-} \mu \not\vdash \mu$ (*success*)

(C4) if $kb_0 \equiv kb_1$ and $\mu_0 \equiv \mu_1$ then $kb_0 \dot{-} \mu_0 \equiv kb_1 \dot{-} \mu_1$ (*preservation*)

(C5) $(kb \dot{-} \mu) \wedge \mu \vdash kb$ (*recovery*)

By *inclusion*, every consequence of $kb \dot{-} \mu$ is also a consequence of kb, i.e. contractions remove but do not add propositions. By *vacuity*, if μ is not a consequence of kb, then $kb \dot{-} \mu$ has all of the original consequences of kb. By *success*, the contraction of kb by μ always excludes μ in all but the limiting case where μ is valid. By *preservation*, contractions on equivalent bases by equivalent propositions are always equivalent. Finally, by *recovery*, if kb is contracted by μ, and then extended by μ, all of the original consequences of kb are recovered.

An important property of Gärdenfors' Postulates is the class of operators they admit. Ideally, the class would have only one member, and therefore contraction and revision would be defined uniquely. In fact, Gärdenfors' Postulates admit a class whose cardinality is the same as the language in which contractions and revisions are expressed. For example, the possible contractions on

$$(a \wedge b) \dot{-} b$$

correspond to every formula implied by $(a \wedge b)$ with consequence $(\neg b \vee a)$ which does not itself imply b. For CPL, this set is countably infinite. One of the problems with axiomatic approaches in general is their limitation as an explanatory vehicle in the absence of analytical models to justify their existence. Arguably, credence to axiomatic properties should not be lent *a priori*, but should reflect an independent semantic account which explicitly identifies the essential features governing an operation.

2.2 (PROPOSITIONAL) KNOWLEDGE-BASE REVISION

A model-theoretic analysis of recent Database and AI proposals for *revision* (Borgida 1985), (Dalal 1988), (Satoh 1988), (Weber 1986), (Winslett 1988) has been provided by Katsuno and Mendelzon (1990a, 1990b). Foremost among the collective approaches used to define $kb\dot{+}\mu$ is the intuition (borrowed from Gärdenfors) that revising kb by μ should involve only the *minimal change* to kb required to accommodate μ. The model-theoretic counterpart to this idea is that the models $\mathcal{M}(kb\dot{+}\mu)$ of $kb\dot{+}\mu$ should be *close* to the models $\mathcal{M}(kb)$ of kb. The analysis of $kb\dot{+}\mu$ provided by Katsuno and Mendelzon (1990a) is based on *pre-orders* $\leq_{kb}$ over $\mathcal{M}(kb)$: given every model of $\mathcal{M}(kb\dot{+}\mu)$ is also a model of $\mathcal{M}(\mu)$, $\mathcal{M}(kb\dot{+}\mu)$ is said to represent the set of models of $\mathcal{M}(\mu)$ which are *minimal* with respect to $\leq_{kb}$, i.e. 'closest' to $\mathcal{M}(kb)$. Classifying models as 'close together' is representative of a distance measure. As a group, therefore, the approaches in this category might be referred to as *distance* or *metric* approaches. For example, Winslett (1988) has proposed a revision operator called the *possible models approach* (PMA), where the distance between two models is measured by the set of propositional letters on which they differ.

One comment that can be made regarding metrics has to do with situations where kb is *unsatisfiable*. By definition, $\mathcal{M}(kb) = \emptyset$ whenever kb is unsatisfiable, i.e. kb has *no* models. According to Winslett's proposal (1988), $\mathcal{M}(kb\dot{+}\mu)$ is defined as the union of the models of $\mathcal{M}(\mu)$ closest to m, for $m \in \mathcal{M}(kb)$. Therefore, $\mathcal{M}(kb\dot{+}\mu) = \emptyset$ whenever $\mathcal{M}(kb) = \emptyset$; that is, $kb\dot{+}\mu$ is unsatisfiable whenever kb is, for arbitrary μ. Thus, it is not possible to revise an inconsistent knowledge-base so that it becomes consistent. Not all metrics are limited in this way. But it does point to a limitation of metric approaches in general: all inconsistent kbs must be treated uniformly (since, in all cases, $\mathcal{M}(kb) = \emptyset$).

However, there are often good reasons for wanting to differentiate between, for example, $(a \wedge \neg a)\dot{+}a$ and $(b \wedge \neg b)\dot{+}a$, even though the unrevised formula in each case is inconsistent. Notably, $(a \wedge \neg a)\dot{+}a$ can be equated with a, since the revision to accept a should at least lose the information about $\neg a$, while $(b \wedge \neg b)\dot{+}a$ might be equated with $(b \wedge \neg b)$, since losing information about $\neg a$ can be considered independent of the information about b or $\neg b$.

To illustrate, consider two universities which use the same DBMS to record student information. Several records are kept on each student, and information is shared among them. For example, 'year-in-progress' is maintained in two separate database records. At the first university, Queen's, a student has her year-in-progress listed once as 1st-year and once as 2nd-year. At the second university, King's, a student has her year-in-progress listed once as 2nd-year and once

as 3rd-year. In both cases, year-in-progress is assumed by the DBMS to be unique. Assuming a classical semantics for the DBMS, the recorded year-in-progress information renders the databases at both Queen's and King's inconsistent.

For recording consistent metric updates $db_Q \dot{+} \mu_Q$ and $db_K \dot{+} \mu_K$ at Queen's and King's, two options present themselves. One is to replace db_Q and db_K by μ_Q and μ_K, respectively, as the updated databases. The other option is to admit a consistent but uniform update. However, in this case, the information expressed by the databases at either Queen's or King's must be exactly the same after the update is performed. The proper updates, therefore, lie outside the model's scope.

2.3 'INFORMATIVE' PARTIAL-ORDERS

In the pre-order account of contraction, $a \leq b$ is interpreted as saying that a is at least as 'close' as b to some semantic object $\mathcal{S}$. However, metrics are about *distance* and not about *information*. Arguably, it is not the metric (or pre-order) which is fundamental to a semantic explanation of revision, but the notion of information which underlies it. Rather than regard '$\leq$' as an ordering of distance, it might instead be regarded as an ordering of information (Scott 1972, 1982). That is, $a \leq b$ would be interpreted as saying that a is at least as *informative*, or at least as good a description of $\mathcal{S}$, as b. What kind of properties should then be expected of '$\leq$'? First, every element should be understood to be at least as informative as itself:

(i) for all x, $x \leq x$ (*reflexive*).

Also, it should be expected that if y is at least as informative as x, and z is at least as informative as y, then z should be at least as informative as x:

(ii) if $x \leq y$ and $y \leq z$, then $x \leq z$ (*transitive*).

Of course, these are just the properties required of a pre-order. What makes orderings of information different than orderings of distance? Because the elements intuitively correspond to 'information', one would expect that if x is at least as informative as y, and y is at least as informative as x, then x and y correspond to the *same* information:

(iii) if $x \leq y$ and $y \leq x$, then $x = y$ (*anti-symmetric*).

A set P together with a binary relation $\leq$ which satisfies (i), (ii), and (iii), for all $x, y, z \in P$, is called a *partially-ordered set* (or *poset*). By definition, partially-ordered sets are also pre-ordered sets. (iii) represents the formal counterpart to the idea that metrics are not about information, even though they may be defined over semantic objects like models. For example, two models x and y can both be the same distance from a model z even though they do not represent the same information.

3 A NEW APPROACH

In this section, a new approach to contraction is presented, where '$\dot{-}$' is regarded as a monotone operator in a language $\mathcal{L}$ complete with its own semantics. Toward this end, let

$$\alpha \dot{-} \beta$$

be interpreted as '*α with no information about β*'.

3.1 PRELIMINARIES

3.1.0 Definition.

(i) If P is a poset then the *least element* $\bot_P$ of P is defined as the least element $z \in P$ s.t. $z \leq x$ for all $x \in P$.

(ii) For sets P, let $P_\bot = (P \cup \{\bot\}, \leq)$, where for all $x, z \in P$, $x \leq z \Rightarrow (x = \bot)$ or $(x = z)$, i.e. $\bot \leq x$ for all $x \in P$, and $x \not\leq z$ whenever $x \neq z$, for $x, z \in P$. $P_\bot$ is called the *flat* poset on P.

(iii) If P is a poset, and $S \subseteq P$, then the *least upper bound* of S, denoted by $\bigsqcup_P S$, is defined as the least element $z \in P$ s.t. $x \leq z$ for all $x \in S$.

Note that $\bigsqcup_P S$ and $\bot_P$ are unique whenever they exist. Subscripts may be omitted whenever there is no ambiguity.

3.1.1 Definition.

(i) A function $f : P \to P'$ between two posets P and P' is said to be *monotone* iff for all $x, x' \in P$,

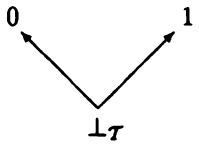
$$x \leq_P x' \text{ implies } f(x) \leq_{P'} f(x').$$

(ii) For two posets P and P', the *montone function space* $[P \to P']$ denotes the set of all *monotone* functions between P and P' with the pointwise order, i.e. for $f, g \in [P \to P']$ and $x \in P$,

$$f \leq_{[P \to P']} g \text{ iff } f(x) \leq_{P'} g(x).$$

3.2 VALUATIONS

3.2.0 Definition. Let $\mathcal{T} = (\{0, 1, \bot_\mathcal{T}\}, \leq_\mathcal{T})$, where $\leq_\mathcal{T}$ is defined by



$\mathcal{T}$ is the flat poset on classical truth-values. Intuitively, a proposition p valuates to 0 whenever it is *false*, valuates to 1 whenever it is *true*, and valuates to $\bot_\mathcal{T}$ whenever there is *no information* about whether p is *true* or *false*.

3.2.1 Definition. Let $\Phi = \{p_0, \ldots, p_k, \ldots\}$ denote a countable set of propositional variables, and let $\mathcal{L}(\Phi)$ be the least set of formulas closed under $\neg$, $\vee$, $\wedge$, $\supset$, and $\dot{-}$. Therefore, whenever α and β are in $\mathcal{L}(\Phi)$, then so are $\neg\alpha$, $\alpha \vee \beta$, $\alpha \wedge \beta$, $\alpha \supset \beta$, and $\alpha \dot{-} \beta$.

Let Φ_x denote any finite set of elements of Φ, where $x \in \mathcal{N}$. A *valuation* is any monotone function $\Phi_x \to \mathcal{T}$. Sets of valuations are ordered pointwise:

3.2.2 Definition. $V_x = [\Phi_x \to \mathcal{T}]$, $\Phi_x \subseteq \Phi$ is finite.

Note that since Φ_x and $\mathcal{T}$ are both finite, V_x is finite.

By definition, for $v, v' \in V_x$, $v \leq_{V_x} v'$ iff for all atomic propositions $p \in \Phi_x$, $v(p) \leq_{\mathcal{T}} v'(p)$. The elements of V_x can be considered 'possible-worlds', with an accessability relation between worlds provided by $\leq_{V_x}$. For example, consider V_x on the two element set given by $\Phi_x = \{a, b\}$ (Figure 1). Accordingly, the set of possible-worlds identified with an arbitrary $v \in V_x$ is obtained by way of its upper set $v\uparrow = \{v' \in V_x \mid v \leq_{V_x} v'\}$. For example, the 'state of the world' given by v_5 is that a is $\perp$ (*no information*), while b is 0 (*false*). The possible-worlds according to v_5, therefore, include v_0, where a and b are both 0 (*false*), and v_2, where a is 1 (*true*) and b is 0 (*false*).

3.2.3 Definition. A valuation $v \in V_x$ is said to be *classical* iff $v(p) \in \{0, 1\}$ for all $p \in \Phi_x$; otherwise v is said to be *non-classical*.

Note that the set of *classical* valuations in V_x correspond to the set of *maximal* (or '*top*') elements in V_x.

3.3 [·]-FORMULA MEANINGS

A *formula meaning* is any monotone function $V_x \to \mathcal{T}$. Sets of formula meanings are also ordered pointwise:

3.3.0 Definition. $M_x = [V_x \to \mathcal{T}]$, $\Phi_x \subseteq \Phi$ is finite.

Note that since V_x and $\mathcal{T}$ are both finite, M_x is finite.

Let $\mathcal{L}(\Phi_x)$ denote the subset of formulas of $\mathcal{L}(\Phi)$ all of whose propositional variables are contained in Φ_x. For notational convenience, the formula meaning associated with an arbitrary formula $\beta \in \mathcal{L}(\Phi_x)$ is denoted by $[\![\beta]\!]_x$; that is,

$$[\![\beta]\!]_x : V_x \to \mathcal{T}.$$

First, the semantics of classical formulas is considered. Initially, the truth-tables for the classical connectives $\neg$, $\vee$, $\wedge$, and $\supset$ are each provided with a monotonic 3-valued analogue. These are defined as follows: [1]

[1] Note that the truth-tables introduced are equivalent to ones presented in a 3-valued logic by S.C. Kleene (1938); see also, Blamey (1986).

3.3.1 Definition.

$\neg$	
1	0
0	1
$\perp$	$\perp$

$\vee$	1	0	$\perp$
1	1	1	1
0	1	0	$\perp$
$\perp$	1	$\perp$	$\perp$

$\wedge$	1	0	$\perp$
1	1	0	$\perp$
0	0	0	0
$\perp$	$\perp$	0	$\perp$

$\supset$	1	0	$\perp$
1	1	0	$\perp$
0	1	1	1
$\perp$	1	$\perp$	$\perp$

The principle which underlies how each of the tables has been defined is to map to the classical truth-values $\{0, 1\}$ whenever a classical interpretation is possible, and to otherwise map to $\perp$ whenever there is ambiguity. Thus, $1 \vee \perp$ maps to 1, since independent of whether $\perp$ were to denote 0 or 1, the value of the expression itself would still map to 1 classically. Similarly, $0 \vee \perp$ maps to $\perp$, since if $\perp$ were to denote 0 the expression would map to 0, while if $\perp$ were to denote 1, the expression would map to 1. Hence, it follows that $\neg\perp$ is $\perp$, while $1 \wedge \perp$ is $\perp$, and $0 \wedge \perp$ is 0.

3.3.2 Definition. For $\alpha, \beta \in \mathcal{L}(\Phi_x)$, and $v \in V_x$,

$$[\![p]\!]_x(v) = v(p), \text{ whenever } p \in \Phi_x$$
$$[\![\neg\alpha]\!]_x(v) = \neg[\![\alpha]\!]_x(v)$$
$$[\![\alpha \vee \beta]\!]_x(v) = [\![\alpha]\!]_x(v) \vee [\![\beta]\!]_x(v)$$
$$[\![\alpha \wedge \beta]\!]_x(v) = [\![\alpha]\!]_x(v) \wedge [\![\beta]\!]_x(v)$$
$$[\![\alpha \supset \beta]\!]_x(v) = [\![\alpha]\!]_x(v) \supset [\![\beta]\!]_x(v)$$

Note that the kind of valuations on classical formulas provided by the Kleene approach are always determined by strictly 'local' considerations. That is, whenever all of the propositional variables in a formula map to $\perp_{\mathcal{T}}$, then the formula itself maps to $\perp_{\mathcal{T}}$, independent of its logical structure. For example, whenever $p \in \Phi_x$ valuates to $\perp_{\mathcal{T}}$, then both classical tautologies and classical inconsistent formulas based on p, such as $(p \vee \neg p)$ and $(p \wedge \neg p)$, also valuate to $\perp_{\mathcal{T}}$.

Each partially-ordered set of formula meanings M_x can be expressed by a truth-table. For example, consider the truth-table generated by some formula meanings of M_x, where $\Phi_x = \{a, b\}$:

	$[\![a]\!]_x$	$[\![b]\!]_x$	$[\![\neg a]\!]_x$	$[\![a \vee b]\!]_x$	$[\![a \wedge b]\!]_x$	$[\![a \supset b]\!]_x$
v_0	0	0	1	0	0	1
v_1	0	1	1	1	0	1
v_2	1	0	0	1	0	0
v_3	1	1	0	1	1	1
v_4	0	$\perp$	1	$\perp$	0	1
v_5	$\perp$	0	$\perp$	$\perp$	0	$\perp$
v_6	$\perp$	1	$\perp$	1	$\perp$	1
v_7	1	$\perp$	0	1	$\perp$	$\perp$
v_8	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

By definition, all formula meanings $m \in M_x$ are *monotone*, and therefore $m(v') \leq_{\mathcal{T}} m(v)$ whenever $v' \leq_{V_x} v$.

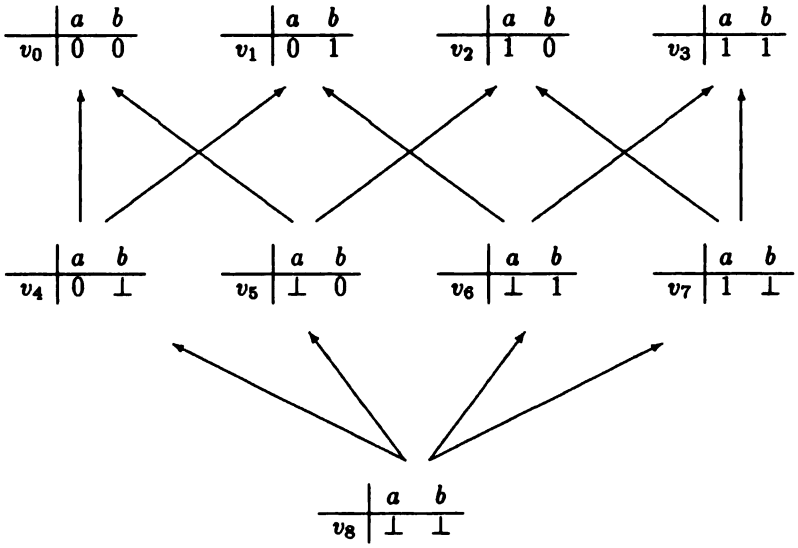


Figure 1: $V_x = \{a, b\}$

Hence, the rows of the truth-table are themselves partially-ordered: the *maximal* ('*top*') elements correspond to the set of classical formula meanings. For example, v_2 denotes a classical valuation, with v_2, v_7, and v_8 partially-ordered as follows:

	$[a]_x$	$[b]_x$	$[\neg a]_x$	$[a \vee b]_x$	$[a \wedge b]_x$	$[a \supset b]_x$
v_2	1	0	0	1	0	0
v_7	1	⊥	0	1	⊥	⊥
v_8	⊥	⊥	⊥	⊥	⊥	⊥

3.3.3 Proposition. For $\alpha, \beta, \lambda \in \mathcal{L}(\Phi_x)$,

(i) $[\neg\neg\alpha]_x = [\alpha]_x$
(ii) $[\alpha \vee (\beta \wedge \lambda)]_x = [(\alpha \vee \beta) \wedge (\alpha \vee \lambda)]_x$
(iii) $[\alpha \wedge (\beta \vee \lambda)]_x = [(\alpha \wedge \beta) \vee (\alpha \wedge \lambda)]_x$
(iv) $[\neg(\alpha \vee \beta)]_x = [\neg\alpha \wedge \neg\beta]_x$
(v) $[\neg(\alpha \wedge \beta)]_x = [\neg\alpha \vee \neg\beta]_x$
(vi) $[\alpha \supset \beta]_x = [\neg\alpha \vee \beta]_x$

3.4 CONTRACTION

For $\alpha, \beta \in \mathcal{L}(\Phi_x)$, information is conveyed about $[\alpha]_x$ and $[\beta]_x$ at each valuation $v \in V_x$, encapsulated by the projections $[\alpha]_x(v)$ and $[\beta]_x(v)$. To obtain a semantics for $[\alpha \dot{-} \beta]_x$, that is, to contract $[\alpha]_x$'s information by $[\beta]_x$, the idea is, for each $v \in V_x$, to evaluate $[\alpha]_x$ not at v, but at some 'less informative' valuation $v' \leq_{V_x} v$ where $[\beta]_x(v') = \perp_{\mathcal{T}}$, i.e. where there is *no information* about $[\beta]_x$. Of course, there may be many worlds $v' \leq_{V_x} v$ where $[\beta]_x(v') = \perp_{\mathcal{T}}$, i.e. where there is no information about $[\beta]_x$. Therefore, $[\alpha \dot{-} \beta]_x(v)$ is evaluated as the *least upper bound* among the elements $[\alpha]_x(v')$, where $v' \leq_{V_x} v$ and $[\beta]_x(v') = \perp_{\mathcal{T}}$. Intuitively, the least upper bound provides a summary of the information about $[\alpha]_x$ encapsulated by v' worlds, which all share the property that none of them provide any information about $[\beta]_x$.

3.4.0 Definition. For $\alpha, \beta \in \mathcal{L}(\Phi_x)$, and $v \in V_x$,

$$[\alpha \dot{-} \beta]_x(v) = \bigsqcup_{\mathcal{T}} \{[\alpha]_x(v') \mid v' \leq_{V_x} v \text{ and } [\beta]_x(v') = \perp_{\mathcal{T}}\}.$$

To illustrate the semantics, consider valuations on $[(a \wedge b) \dot{-} b]_x$, presented in the table below. Note that valuations can be determined tracing along the partial-order outlined in Figure 1, e.g. $[(a \wedge b) \dot{-} b]_x(v_2) = \perp_{\mathcal{T}}$ is obtained by noting that v_7 is a maximal element below v_2 where $[b]_x$ valuates to $\perp_{\mathcal{T}}$ and $[a \wedge b]_x(v_7) = \perp_{\mathcal{T}}$.

	$[a]_x$	$[b]_x$	$[a \wedge b]_x$	$[(a \wedge b) \dot{-} b]_x$
v_0	0	0	0	0
v_1	0	1	0	0
v_2	1	0	0	⊥
v_3	1	1	1	⊥
v_4	0	⊥	0	0
v_5	⊥	0	0	⊥
v_6	⊥	1	⊥	⊥
v_7	1	⊥	⊥	⊥
v_8	⊥	⊥	⊥	⊥

According to the semantics, $[\alpha \dot{-} \beta]_x \leq_{M_x} [\alpha]_x$; under $\leq_{M_x}$, therefore, $[(a \wedge b) \dot{-} b]_x$ is no more informative than $[a]_x$.

3.4.1 Proposition. For $\alpha, \beta \in \mathcal{L}(\Phi_x)$,

$[\alpha \dot{-} \beta]_x : V_x \to \mathcal{T}$ is *monotone*, i.e. $[\alpha \dot{-} \beta]_x \in M_x$.

Finally, some general properties about '$\dot{-}$' are presented. These will be useful when semantic consequence is considered in 3.6. For $\varphi \in \mathcal{L}(\Phi_x)$, let Φ_φ denote exactly the set of propositional variables of Φ_x which appear in φ.

3.4.2 Proposition. For $\varphi, \psi, \lambda \in \mathcal{L}(\Phi_x)$, and $v \in V_x$,

(i) $[\varphi \dot{-} \varphi]_x = \bot_{M_x}$
(ii) $[\varphi \dot{-} \lambda]_x \leq_{M_x} [\varphi]_x$
(iii) $[\varphi \dot{-} (\lambda \dot{-} \lambda)]_x = [\varphi]_x$
(iv) $[\varphi \dot{-} \lambda]_x = [(\varphi \dot{-} \lambda) \dot{-} \lambda]_x$
(v) $[\varphi \dot{-} \lambda]_x = [\varphi]_x$, *whenever* $\Phi_\varphi \cap \Phi_\lambda = \emptyset$
(vi) $[\neg(\varphi \dot{-} \lambda)]_x = [\neg\varphi \dot{-} \lambda]_x$
(vii) $[(\varphi \vee \psi) \dot{-} \lambda]_x(v) = [\varphi \dot{-} \lambda]_x(v) \vee [\psi \dot{-} \lambda]_x(v)$
(viii) $[(\varphi \wedge \psi) \dot{-} \lambda]_x(v) = [\varphi \dot{-} \lambda]_x(v) \wedge [\psi \dot{-} \lambda]_x(v)$
(ix) $[\varphi \dot{-} \lambda]_x = [\varphi \dot{-} \neg\lambda]_x$

3.5 $\mathcal{C}[\cdot]$-FORMULA MEANINGS

It is notable that classical consequence '$\models$' does not correspond to '$\leq_{M_x}$'; for example, although $(a \wedge b) \models a$, $[a]_x \not\leq_{M_x} [a \wedge b]_x$. Also, while classical formulas map to classical truth-values at classical valuations, this is not true for non-classical formulas, e.g. by 3.4.2(i), $[\varphi \dot{-} \varphi]_x(v) = \bot_{\mathcal{T}}$ for all $v \in V_x$. Hence, $\models$ cannot be applied to $[\cdot]_x$-formula meanings directly to obtain a definition of semantic consequence; nor is $\leq_{M_x}$ a generalization which presents a conservative extension to CPL. The next topic to be considered is how to 'recover' classical $\models$ for non-classical elements in M_x.

3.5.0 Definition. A formula $\alpha \in \mathcal{L}(\Phi_x)$ is said to be *unsatisfiable* at $v \in V_x$ whenever $[\alpha]_x(v) = 0$; otherwise, α is said to be *satisfiable* at v. Hence, α is *satisfiable* at v whenever α is not *false*, i.e. $[\alpha]_x(v) \neq 0$; or equivalently, $[\alpha]_x(v) \in \{1, \bot_{\mathcal{T}}\}$. Finally, α is said to be *valid* whenever α is *satisfiable* at v, for all $v \in V_x$.

Note that $[\alpha]_x(\bot_{V_x}) = \bot_{\mathcal{T}}$, for all $\alpha \in \mathcal{L}(\Phi_x)$, and so there always exists a world $v \in V_x$ for every $\alpha \in \mathcal{L}(\Phi_x)$ such that α is *satisfiable* at v [2]. However, not all formulas are *classically satisfiable*; that is, satisfiable at some $v \in V_x$ where v is *classical*. For example, although $[p \wedge \neg p]_x$ and $[p \vee \neg p]_x$ are mapped 'locally' by $\bot_{V_x}$ to $\bot_{\mathcal{T}}$, it is evident that in every world v where there is 'more information' about p, e.g. $\bot_{V_x} \leq_{V_x} v$ and $v(p) = 0$ or $v(p) = 1$, that $[p \wedge \neg p]_x(v) = 0$, while $[p \vee \neg p]_x(v) = 1$. Of course, $(p \vee \neg p)$ is *valid*.

[2] Note that $\mathcal{L}(\Phi_x)$ has no constants for 1 and 0.

3.5.1 Definition. Let '$\approx$' be an *equivalence relation* on M_x, s.t. for $\rho, \rho' \in M_x$, $\rho \approx \rho'$ iff for all $v \in V_x$,

$$\rho(v) \in \{1, \bot_{\mathcal{T}}\} \Leftrightarrow \rho'(v) \in \{1, \bot_{\mathcal{T}}\}.$$

For $\alpha \in \mathcal{L}(\Phi_x)$, the *equivalence class* $[\alpha]_{\mathcal{C}}$ generated by $[\alpha]_x$ is called the *$\mathcal{C}$-class of α*.

The interesting property about $\mathcal{C}$-classes is that whenever $\rho \approx \rho'$ then ρ is satisfiable at $v \in V_x$ iff ρ' is satisfiable at v. For example, $[p \dot{-} p]_x \approx [p \vee \neg p]_x$. Also, $[(a \wedge b) \dot{-} b]_x \approx [a]$; hence, a is satisfiable at $v \in V_x$ whenever $(a \wedge b) \dot{-} b$ is, and vice versa. In the classical model, tautologies are assigned to the same truth-value at every valuation. Hence, no semantic basis exists by which to distinguish them. This property has an analogue under $\mathcal{T}$: all tautologies are assigned by '$\approx$' to the same $\mathcal{C}$-class.

However, although $(p \dot{-} p)$ and $(p \vee \neg p)$ belong to the same $\mathcal{C}$-class, $\neg(p \dot{-} p)$ and $\neg(p \vee \neg p)$ do not. Similarly, $\neg[(a \wedge b) \dot{-} b]$ and $\neg a$ do not belong to the same $\mathcal{C}$-class. In fact, $\neg(p \dot{-} p)$, $\neg[(a \wedge b) \dot{-} b]$, and $[(a \wedge b) \dot{-} b] \supset b$ are all valid. This can be viewed as a general property of non-classical formulas in $\mathcal{L}(\Phi_x)$: their classical negation and implication structure is vacuous. To obtain a semantics where it is possible to contract classical inconsistent formulas such as $(p \wedge \neg p)$, there must exist a valuation $v \in V_x$ where $[p \wedge \neg p]_x(v) = \bot_{\mathcal{T}}$, i.e. $[p]_x(v) = \bot_{\mathcal{T}}$. But then, $\bot_{\mathcal{T}} \wedge \neg\bot_{\mathcal{T}} = \bot_{\mathcal{T}}$, and so $\neg\bot_{\mathcal{T}} = \bot_{\mathcal{T}}$. However, if negation is to operate this way, then little should be expected from contradiction arguments based upon a formula and its negation, which is exactly the semantic justification for classical implication.

Even though $\neg[p \dot{-} p]$ is valid, every element has a complement in M_x. Formally,

3.5.2 Definition. For $\rho \in M_x$, let $-\rho$ be the *least* element $\neg\rho \leq_{M_x} -\rho$ s.t. for *classical* $v \in V_x$,

$$\rho(v) \vee -\rho(v) = 1 \text{ and } \rho(v) \wedge -\rho(v) = 0.$$

$-\rho$ is said to be the *complement* of ρ in M_x.

Note that $-[\alpha] = \neg[\alpha]$ whenever α is classical; also, $--\rho \approx \rho$, for $\rho \in M_x$. This is reminiscent of the classical property that $\neg\neg p \equiv p$. $(p \dot{-} p)$'s complement may be obtained as follows: for $v \in V_x$, let

$$\bot_\Delta(v) = \begin{cases} 1 & \text{if } v \in V_x \text{ is classical} \\ \bot_{\mathcal{T}} & \text{otherwise.} \end{cases}$$

Then, $-[p \dot{-} p]_x = \neg\bot_\Delta$. Note, for $\Phi_x = \{p_0 \ldots p_n\}$,

$$\bot_\Delta = [(p_0 \vee \neg p_0) \wedge \ldots \wedge (p_n \vee \neg p_n)]_x.$$

$\neg\bot_\Delta$ is the complement of $[p \dot{-} p]_x$ and $\bot_\Delta$ in M_x, although only $\bot_\Delta$ is the complement of $\neg\bot_\Delta$. Hence, $--[p \dot{-} p] = \bot_\Delta$, e.g. $--[p \dot{-} p]$ is *classical*, and

$$[p \dot{-} p] <_{M_x} --[p \dot{-} p]_x.$$

Like all $\mathcal{C}$-classes, $[p \dot{-} p]_{\mathcal{C}}$ corresponds to a lattice in M_x; the classical tautologies represent a lattice interior to $[p \dot{-} p]_{\mathcal{C}}$. $\bot_\Delta$ is the least element in this sub-lattice.

Alternatively, $\perp_\Delta$ expresses as the least element among those in $[p \dot{-} p]_C$, where, for *classical* $v \in V_x$, $\perp_\Delta(v) \in \{0,1\}$, i.e. the least element $[\![p \dot{-} p]\!]_x \leq_{M_x} \perp_\Delta$ in $[p \dot{-} p]_C$ which 'recovers' properties of classical negation. Note that a similar property is extended to $[\![(a \wedge b) \dot{-} b]\!]_x$ in $[(a \wedge b) \dot{-} b]_C$, e.g. $-\!-[\![(a \wedge b) \dot{-} b]\!]_x = [\![a]\!]_x \wedge \perp_\Delta$, and $[\![a]\!] \wedge \perp_\Delta$ is *classical*.

3.5.3 Definition. For $\rho \in M_x$, let $\Delta\rho$ be the *least* element $\rho \leq_{M_x} \Delta\rho$ s.t. $\rho \approx \Delta\rho$; and,

$$\Delta\rho(v) \in \{0,1\}, \text{ for } \textit{classical } v \in V_x.$$

$\Delta\rho$ is called the *C-class representative* of ρ in M_x.

Note that $-\rho = \neg\Delta\rho$; or equivalently, $-\!-\rho = \Delta\rho$. By construction, Δ recovers *classical* formula meanings from *non-classical* ones, e.g. for $v \in V_x$,

$$\Delta[\![(a \wedge b) \dot{-} b]\!]_x(v) = [\![a]\!]_x(v) \wedge \perp_\Delta(v).$$

Hence, $\Delta[\![(a \wedge b) \dot{-} b]\!]_x$ recovers $[\![a]\!]$ up to ismorphism on classical tautologies. For *classical* valuations $v \in V_x$, $\Delta[\![\alpha]\!]_x(v) = 0$ whenever $[\![\alpha]\!]_x(v) = 0$, and $\Delta[\![\alpha]\!]_x(v) = 1$ whenever $[\![\alpha]\!]_x(v) = 1$ or $[\![\alpha]\!]_x(v) = \perp_{\mathcal{T}}$. Hence, $\Delta[\![\alpha]\!] = [\![\alpha]\!]$ whenever α is *classical*; and, in general, $\neg\Delta[\![\alpha]\!]_x \neq \Delta\neg[\![\alpha]\!]_x$ only if α is *non-classical*.

Parallel to 3.3.2, but for C-class representatives, let '$C[\![\cdot]\!]_x$' extend '$[\![\cdot]\!]_x$' as follows:

3.5.4 Definition. For $\alpha, \beta \in \mathcal{L}(\Phi_x)$, and $v \in V_x$,

$$C[\![p]\!]_x(v) = v(p), \textit{ whenever } p \in \Phi_x$$
$$C[\![\neg\alpha]\!]_x(v) = \neg C[\![\alpha]\!]_x(v)$$
$$C[\![\alpha \vee \beta]\!]_x(v) = C[\![\alpha]\!]_x(v) \vee C[\![\beta]\!]_x(v)$$
$$C[\![\alpha \wedge \beta]\!]_x(v) = C[\![\alpha]\!]_x(v) \wedge C[\![\beta]\!]_x(v)$$
$$C[\![\alpha \supset \beta]\!]_x(v) = C[\![\alpha]\!]_x(v) \supset C[\![\beta]\!]_x(v)$$
$$C[\![\alpha \dot{-} \beta]\!]_x(v) = \Delta[\![\alpha \dot{-} \beta]\!]_x(v)$$

3.5.5 Proposition. For $\alpha \in \mathcal{L}(\Phi_x)$,

$C[\![\alpha]\!]_x : V_x \to \mathcal{T}$ is *monotone*, i.e. $C[\![\alpha]\!]_x \in M_x$.

To recover classical semantic consequence, $\models$ shall be defined over $C[\![\cdot]\!]_x$ (versus $[\![\cdot]\!]_x$).

Recall $[\![\neg(\varphi \dot{-} \lambda)]\!]_x = [\![\neg\varphi \dot{-} \lambda]\!]_x$ by 3.4.2 (vi). However,

$$C[\![\neg(\varphi \dot{-} \lambda)]\!]_x \neq C[\![\neg\varphi \dot{-} \lambda]\!]_x.$$

For example, $C[\![\neg a \dot{-} a]\!]_x$ belongs to the C-class of all tautologies, while $C[\![\neg(a \dot{-} a)]\!]_x$ belongs to a C-class of classically unsatisfiable formulas. Of course, this is necessary in order to obtain the required 'fix' for *modus ponens* (3.5.2).

3.6 SEMANTIC CONSEQUENCE

3.6.0 Definition. For $\Gamma \subseteq \mathcal{L}(\Phi_x)$ and $\varphi \in \mathcal{L}(\Phi_x)$,

$\Gamma \models \varphi$ iff for all *classical* valuations $v \in V_x$,

$C[\![\gamma]\!]_x(v) = 1$, for all $\gamma \in \Gamma \Rightarrow C[\![\varphi]\!]_x(v) = 1$.

$\langle \mathcal{L}(\Phi), \models \rangle$ is called the logic C of *contraction*. Notably, '$\models$' considers only *classical* valuations, i.e. only '$\dot{-}$' has access to *non-classical* members in V_x. Hence, by 3.6.0,

3.6.1 Proposition.

C is a *conservative extension* of CPL. □

3.6.2 Proposition. For $\varphi, \psi, \lambda \in \mathcal{L}(\Phi_x)$,

(i) $\models \varphi \dot{-} \varphi$
(ii) $\varphi \models \varphi \dot{-} \lambda$
(iii) $\varphi \dot{-} (\lambda \dot{-} \lambda) \models \varphi$
(iv) $(\varphi \dot{-} \lambda) \dot{-} \lambda \models \varphi \dot{-} \lambda$
(v) $(\varphi \dot{-} \lambda) \wedge \varphi \dot{-} (\varphi \dot{-} \lambda) \models \varphi$
(vi) $\models (\varphi \wedge \psi) \dot{-} \varphi \dot{-} \psi$
(vii) $\varphi \dot{-} \lambda \models \varphi$, *whenever* $\Phi_\varphi \cap \Phi_\lambda = \emptyset$
(viii) $\neg(\varphi \dot{-} \lambda) \models \neg\varphi \dot{-} \lambda$
(ix) $\models (\varphi \vee \psi) \dot{-} \lambda \equiv (\varphi \dot{-} \lambda) \vee (\psi \dot{-} \lambda)$
(x) $\models (\varphi \wedge \psi) \dot{-} \lambda \equiv (\varphi \dot{-} \lambda) \wedge (\psi \dot{-} \lambda)$
(xi) $\models (\varphi \dot{-} \lambda) \equiv (\varphi \dot{-} \neg\lambda)$
(xii) $\models \lambda \dot{-} (\varphi \wedge \psi) \equiv$
$(\lambda \dot{-} \varphi) \dot{-} \psi \wedge (\varphi \supset (\lambda \dot{-} \psi)) \wedge (\psi \supset (\lambda \dot{-} \varphi))$
(xiii) $\models \lambda \dot{-} (\varphi \vee \psi) \equiv$
$(\lambda \dot{-} \varphi) \dot{-} \psi \wedge (\neg\varphi \supset (\lambda \dot{-} \psi)) \wedge (\neg\psi \supset (\lambda \dot{-} \varphi))$

By (i), $(\varphi \dot{-} \varphi)$ is *valid* in C for all $\varphi \in \mathcal{L}(\Phi_x)$. (ii) shows that contractions are always 'less informative' than their uncontracted counterparts. By (iii), contracting the least informative element is an identity operation, and (iv) shows that $\dot{-}$ is a fixpoint operator in M_x. (v) constitutes C's axiom of *recovery*: $\varphi \dot{-} (\varphi \dot{-} \lambda)$ can be regarded as expressing the information contracted from φ by λ. By (vi), the information expressed by any conjunction is removed by first contracting it by one of its conjuncts, and then the other.

By (vii), if two formulas share no propositional variables, then the contraction of either one by the other is an identity operation. (viii) expresses the left distribution law for '$\neg$'. Note that its converse does not hold. By (ix), (x), (xii) and (xiii), $\dot{-}$ distributes over $\vee$ and $\wedge$ on both the left- and right-hand sides. Hence, a completeness result can be obtained for the operator by providing an algorithm for 'computing' the classical formula to which a contraction corresponds (Lownie 1992). (xi) runs counter to Gärdenfors' Postulate (C2) *vacuity*, with semantic justification: if one has no information about λ, then one has no information about $\neg\lambda$; hence, contraction by a formula or its negation is equivalent under $\models$.

As shown by the example involving inconsistent formulas, classically equivalent propositions do not, in general, determine equivalent contractions. Additionally,

propositional variables cannot be substituted by well-formed formulas to obtain equivalent contractions. For example, while $(a \wedge b) \dot{-} b \models a$,

$$(a \wedge a) \dot{-} a \not\models a.$$

That is, a cannot be substituted for b in $(a \wedge b) \dot{-} b$.

4 A COMPARATIVE ANALYSIS

Briefly, an analysis of C is presented, comparing it to other work.

4.1 INCONSISTENT BASES, RECOVERY

By 3.6.2 (vii), $(\varphi \dot{-} \lambda) \models \varphi$ whenever $\Phi_\varphi \cap \Phi_\lambda = \emptyset$; that is, whenever φ and λ share no propositional letters. Consequently, distinctions emerge in C among contractions to *inconsistent* bases. For example, while $(a \wedge \neg a) \dot{-} b$ is *inconsistent*, $(b \wedge \neg b) \dot{-} b$ is *valid*:

	$C[\![a]\!]_x$	$C[\![b]\!]_x$	$C[\![(a \wedge \neg a) \dot{-} b]\!]_x$	$C[\![(b \wedge \neg b) \dot{-} b]\!]_x$
v_0	0	0	0	1
v_1	0	1	0	1
v_2	1	0	0	1
v_3	1	1	0	1
v_4	0	$\perp$	0	$\perp$
v_5	$\perp$	0	$\perp$	$\perp$
v_6	$\perp$	1	$\perp$	$\perp$
v_7	1	$\perp$	0	$\perp$
v_8	$\perp$	$\perp$	$\perp$	$\perp$

Hence, losing information about b is considered independent of the information about a or $\neg a$. Also, a is equivalent to $[a \wedge (b \wedge \neg b)] \dot{-} b$; and

$$[(a \wedge \neg a) \wedge (b \wedge \neg b)] \dot{-} b \models (a \wedge \neg a),$$

although $[(a \wedge \neg a) \wedge (b \wedge \neg b)] \dot{-} a \dot{-} b$ is *valid*.

As the previous example has shown, $((b \wedge \neg b) \dot{-} b) \wedge b \not\models (b \wedge \neg b)$, and so Gärdenfors' *recovery* Postulate (C5) is not validated by C. Instead, C's *recovery axiom* is expressed by 3.6.2 (v):

$$(\varphi \dot{-} \lambda) \wedge \varphi \dot{-} (\varphi \dot{-} \lambda) \models \varphi.$$

Note that $(\varphi \dot{-} \lambda)$ expresses the information available to φ after contracting by λ, while $\varphi \dot{-} (\varphi \dot{-} \lambda)$ expresses the information actually contracted from φ by λ. For example, $(a \wedge b) \dot{-} b \models a$, and

$$(a \wedge b) \dot{-} ((a \wedge b) \dot{-} b) \models b.$$

Similarly, $(a \wedge \neg b) \dot{-} \neg b \models a$, and

$$(a \wedge \neg b) \dot{-} ((a \wedge \neg b) \dot{-} \neg b) \models \neg b.$$

Also, where φ is $(b \wedge \neg b)$ and λ is b,

$$(b \wedge \neg b) \dot{-} ((b \wedge \neg b) \dot{-} b) \models (b \wedge \neg b).$$

Hence, C admits its own unique version of *recovery*.

4.2 UPDATE AND ERASURE

Consider an example presented by Katsuno and Mendelzon (1990b):

4.2.0 Example. Let $kb \equiv (a \wedge \neg b) \vee (\neg a \wedge b)$ and assume $\mu \equiv a$. Now, suppose a room contains a table and two objects, a book and a magazine. Let a be interpreted '*the book is on the floor*', and b, '*the magazine is on the floor*'. Then, kb states that either the book is on the floor, or the magazine is on the floor, but not both. A person is then instructed to enter the room and ensure the book is placed on the floor. The new state of the world, therefore, corresponds to $kb \dot{+} \mu$. Since kb is consistent with μ, by Gärdenfors' Postulate (R2), the revision should be equivalent to $kb \wedge \mu$, or $a \wedge \neg b$. But why is it reasonable to assume that the magazine is not on the floor?

According to Katsuno and Mendelzon, this example helps to illustrate the difference between *belief revision* and *database update*. In their view, *belief revision* has to do with fixing inconsistent theories; that is, correcting a theory kb so that it expresses an accurate picture of the world consistent with what is known about it. Conversly, *database update* is regarded as recording changes which occur in a world; for example, having the book placed on the floor. In this case, kb should not be interpreted as incorrect with respect to what it has been intended to represent, but outdated with respect to a change which has occurred in the world. Hence, 'update' refers to the operation of having this change reflected in kb. By this argument, (R2) is not an appropriate assumption to make regarding updates, because the world under consideration changes each time.

Two sets of postulates, alternatives to those presented by Gärdenfors for *revision* and *contraction*, are presented by Katsuno and Mendelzon (1991b) for a new class of operators called *update* and *erasure*. For example, the *update* postulate corresponding to (R2) is

(U2) if kb implies μ then $kb \dot{+} \mu$ is equivalent to kb

Note that (U2) is weaker than its corresponding revision postulate (R2): it says that whenever μ can be obtained from kb then $kb \dot{+} \mu$ is the same as kb, while (R2) says that whenever $\{kb, \mu\}$ is satisfiable, then $kb \dot{+} \mu$ is the same as $kb \wedge \mu$. For example, Winslett's PMA operator (1988) provides the intuitively correct result for Example 4.2.0 (i.e. $kb \dot{+} a$ is equivalent to a), and therefore would be classified as an *update* rather than a *revision* operator.

It is worthwhile noting that (U2) has an important consequence: whenever kb is inconsistent, then $kb \dot{+} \mu$ is inconsistent for any μ. Notably, this should be regarded as another illustration of the difference between update and revision. An inconsistent knowledge-base is the result of an inadequate theory; updates are about recording changes in a world. By definition, an

inconsistent knowledge-base describes *no* world and so there can be no change to a world which corresponds to it. Hence, there is no update to record it.

The problem with this view is that neither revision nor update alone is considered adequate to handle all of the changes required for a knowledge-base. Instead, a hybrid approach must be advocated.

Consider a variation on Example 4.2.0:

4.2.1 Example. Let $kb \equiv (a \wedge \neg b) \vee (\neg a \wedge b)$ and assume $\mu \equiv a$. Now, suppose a train is traveling westward toward a forked juncture. Let a be interpreted '*the fork is switched to the left*', and b, '*the fork is switched to the right*'. Then, kb states that either the fork is switched to the left, or the fork is switched to the right, but not both. A train engineer is then instructed to ensure the fork is switched to the left. The new state of the world, therefore, corresponds to $kb\dot{+}\mu$. Since kb is consistent with μ, by Gärdenfors' Postulate (R2), the revision should be equivalent to $kb \wedge \mu$, or $a \wedge \neg b$. However, switching the fork to the left corresponds to an event occurring in the world, and so the change should be recorded by *update* rather than *revision*. However, *updates* do not guarantee $a \wedge \neg b$. But why is it not reasonable to assume that the fork is not switched to the right?

Example 4.2.1 illustrates one case where *revision* guarantees the proper update while *update* does not, even though the situation has to do with recording a change in the world rather than fixing an inconsistent theory. Alternatively, the semantics for $\mathcal{C}$ refer only to *changing information*. No epistemic position has been taken with respect to whether 'information' refers to beliefs or to events in a world. Is it possible to apply $\mathcal{C}$'s semantics uniformly to $kb\dot{+}\mu$ (and $kb\dot{-}\mu$) without regard to epistemic interpretations on kb or μ ?

First of all, note that the semantics for $\mathcal{C}$ provide the same answer to Example 4.2.0 as Winslett's PMA operator; that is, $[(a \wedge \neg b) \vee (\neg a \wedge b)]\dot{+}a$ is equivalent to a. Hence, $\dot{+}$ does not validate Gärdenfors' Postulate (R2). However, $\dot{+}$ also does not validate (U2), as the example involving *inconsistent* bases has shown. Therefore, $\mathcal{C}$'s version of '$\dot{+}$' is neither a *revision* operator, in the sense of Gärdenfors (1985), nor is it an *update* operator in the sense of Katsuno and Mendelzon (1991b).

Of course, there is still the problem of how to model both Example 4.2.0 and Example 4.2.1 relying on only a single operator '$\dot{+}$'. At first glance, it looks as though two separate updates are required of the same operation

$$(a \equiv b)\dot{+}a,$$

depending on what interpretation is given to a and b. However, the problem becomes easier if it is recognized that while

$$(\textit{book-on-floor} \equiv \neg \textit{magazine-on-floor})$$

can be considered a statement *contingently true*, and therefore subject to change,

$$(\textit{fork-set-to-left} \equiv \neg \textit{fork-set-to-right})$$

can be considered a statement which is *true by definition*, or an *integrity constraint*. That is, it is never the case that the fork can be set both to the left and the right, or set neither to the left or to the right.

Integrity constraints have priviledged status within a database system because they are considered immune to database updates. Hence, they are statements expressed by a database which are *defined* to always be true. Informally, they might be ascribed to have the same information as the tautologies (or *valid* sentences) underlying a logic. For example, assume $\mathcal{C}$ 'database queries' have the form:

$$\models kb \supset \alpha.$$

To express β as a database integrity constraint, the idea is to include it alongside the set of tautologies which are implicitly represented on the left-hand side of '$\models$'; that is,

$$\beta \models kb \supset \alpha.$$

In this way, β remains independent of all operations on kb involving '$\dot{-}$' or '$\dot{+}$'; for example,

$$\beta \models (kb \dot{-} \mu) \supset \beta, \text{ for all } \beta, \mu \in \mathcal{L}(\Phi_x).$$

By way of integrity constraints, Examples 4.2.0 and 4.2.1 can be modeled in $\mathcal{C}$ as follows: for 4.2.0,

$$\models kb\dot{+}a$$

validates a, but not $a \wedge \neg b$. However, in the case of 4.2.1,

$$(a \wedge \neg b) \vee (\neg a \wedge b) \models kb\dot{+}a$$

validates $a \wedge \neg b$ for arbitrary propositions kb.

4.3 GÄRDENFORS' LEMMA

Consider Katsuno and Mendelzon's Postulate (U8) for *disjunction* (1990b):

(U8) $(kb_0 \vee kb_1)\dot{+}\mu \equiv (kb_0\dot{+}\mu) \vee (kb_1\dot{+}\mu)$

It says that updating two knowledge-bases kb_0 and kb_1 by μ, and then obtaining what is common between them, is the same as first obtaining what is common between them, and then updating this by μ. (U8) validates an interesting *monotonicity* property:

4.3.0 Proposition. If an update operator $\dot{+}$ satisfies (U8) then

$$(kb_0\dot{+}\mu) \vdash (kb_1\dot{+}\mu) \text{ whenever } kb_0 \vdash kb_1$$
$$(\textit{monotonicity}).$$

The CPL proof is as follows. Assume (U8), and let $kb_0 \vdash kb_1$. Then,

$$kb_1 \dot{+} \mu \equiv (kb_0 \vee kb_1) \dot{+} \mu$$
$$\equiv (kb_0 \dot{+} \mu) \vee (kb_1 \dot{+} \mu),$$

that is, $(kb_0 \dot{+} \mu) \vdash (kb_1 \dot{+} \mu)$. □

This property forms the basis of Gärdenfors' '*impossibility lemma*' (1988), which shows that there is no non-trival revision operator which satisfies both *monotonicity* and Gärdenfors' Postulates (R1)-(R4) [3].

A curious property of $\mathcal{C}$'s version of '$\dot{+}$' is that it validates (U8), but as the examples involving *inconsistent bases* show, it does not validate *monotonicity*. Yet, this would seem to violate Proposition 4.3.0 above. In CPL, whenever $kb_0 \models kb_1$, then kb_1 and $(kb_0 \vee kb_1)$ denote the *same* formula meaning. In $\mathcal{C}$'s poset model, they do not, even though it is still the case that kb_0 and $kb_0 \vee kb_1$ are equivalent under '$\models$'; that is,

$$kb_0 \vee kb_1 \models \mu \textit{ iff } kb_1 \models \mu, \text{ for arbitrary } \mu.$$

Hence, the semantics support the (U8) distribution law, without validating *monotonicity*. Aside from other considerations, this property makes it impossible to provide any meaningful semantic analysis of revision to inconsistent bases, since any revision of an inconsistent proposition kb by μ must imply every other revision $kb' \dot{+} \mu$ of every other proposition kb' by μ.

4.4 'INFORMATION GAIN'

In general, classical equivalence is not preserved by '$\dot{-}$'. Notably, this property extends beyond *inconsistent bases*. For example, in $\mathcal{C}$,

$$(a \wedge b) \dot{-} a \not\equiv [a \wedge (a \supset b)] \dot{-} a.$$

Of course, $a \wedge (a \supset b)$ and $(a \wedge b)$ are classically equivalent. The only difference in their formula meanings occurs where $[\![a]\!](v) = \bot_{\mathcal{T}}$ and $[\![b]\!](v) = 0$. Then, $[\![a \wedge (a \supset b)]\!]_x(v) = \bot_{\mathcal{T}}$, while $[\![a \wedge b]\!]_x(v) = 0$. However, this difference becomes more pronounced when each is contracted by a: $(a \wedge b) \dot{-} a$ is equivalent to b, while $(a \wedge (a \supset b)) \dot{-} a$ is equivalent to $(a \dot{-} a)$, i.e. it is *valid*.

An informal reading of this example is that $\dot{-}$ 'sees' b in $(a \wedge b)$, but not in $a \wedge (a \supset b)$, i.e. $\dot{-}$ does not make *implicit* information *explicit* prior to contracting a proposition. This is evident given how $\dot{-}$ operates on *inconsistent* bases: if the operator were based on the consequences of a formula, then all inconsistent bases would have to be treated uniformly. The imposition of *supervaluations* (Van Frassen 1966) would force $[\![(a \wedge b)]\!]_x$ and $[\![a \wedge (a \supset b)]\!]_x$ to be equated in M_x. In this approach, α is assigned to 0 (resp. 1) at v if among the set of classical worlds possible for v, α is always assigned 0 (resp. 1). By definition, supervaluations assign inconsistent classical formulas such as $(a \wedge \neg a)$ to 0, i.e. there is always information about them at every 'world'. Accordingly, there is no way to contract an inconsistent proposition since there is no world v available in V_x where $(a \wedge \neg a)$ is mapped to $\bot_{\mathcal{T}}$, i.e. there is no way to 'lose' this kind of information. Also, the expectation should now be that computing the contraction will require more work since $\dot{-}$ must assume an inference component. And, indeed, this is the case. For example, left distribution by '$\wedge$' (3.6.2(x)) is no longer valid; that is,

$$[a \wedge (a \supset b)] \dot{-} a \not\equiv (a \dot{-} a) \wedge (a \supset b) \dot{-} a.$$

Instead, a more general set of properties aimed at enumerating the consequences of a contraction must be adopted rather than a set of distribution properties to construct it (Lownie 1992).

A problem often cited in connection to Kripke-style possible-worlds semantics is *logical omniscience*: every agent's 'state of knowledge' is closed under logical consequence. Hence, if an agent is said to know a and $(a \supset b)$, it is assumed the agent also knows b, and every other proposition which is a consequence of a and b. However, this view does not account for computational limitations which might exist for an agent or for computational variance between agents. One way to interpret this problem is that the inference from a and $(a \supset b)$ to b is not regarded as contributing additional information to an agent's state of knowledge. When judged from this standpoint,

$$a \wedge (a \supset b) \quad \text{and} \quad a \wedge (a \supset b) \wedge b$$

are identical. However, this is not the case in $\mathcal{C}$: $a \wedge (a \supset b)$ is strictly less informative in M_x than $a \wedge (a \supset b) \wedge b$; or equivalently, $a \wedge b$:

$$[\![a \wedge (a \supset b)]\!]_x <_{M_x} [\![a \wedge b]\!]_x.$$

Hence, the inference from $a \wedge (a \supset b)$ to $(a \wedge b)$ describes real information gain, even though the two lead to the same set of consequences under $\models$. Also, this view of information gain is 'graded' in M_x:

$$[\![a \wedge (a \supset b) \wedge (b \supset c)]\!]_x <_{M_x} [\![a \wedge b \wedge (b \supset c)]\!]_x <_{M_x} [\![a \wedge b \wedge c]\!]_x$$

Note that even though $a \wedge b \models a$, $[\![a]\!]_x \not\leq_{M_x} [\![a \wedge b]\!]_x$, and so $\models$ does not coincide with $\leq_{M_x}$. Instead, $\varphi \models \psi$ whenever

$$[\![\psi \wedge \varphi]\!]_x \leq_{M_x} [\![\varphi]\!]_x.$$

In $\mathcal{C}$, therefore, there is a perspective by which *contraction* can be viewed as an *information loss* operation, and *inference* as an *information gain* operation. Hence, logical omniscience is not built into the model; instead, the model reflects some limitations on computational resources.

[3] A revision operator '$\dot{+}$' is called *trivial* for kb whenever $(kb \dot{+} \mu) \equiv \mu$ for all μ.

4.5 RELEVANCE

Among Anderson and Belnap's (1977) proposals on *relevance*, the logic $\mathcal{R}$ has been reviewed by Belnap (1977) as one with possible benefits to Computer Science. Some notable properties of $\mathcal{R}$ include:

(i) $a \wedge \neg a \not\models b$ (ii) $a \wedge (a \supset b) \not\models b$ (iii) $a \not\models b \vee \neg b$

By (i), classical inconsistencies do not relevantly imply all formulas; and, by (ii), not all classical implications are relevant ones. By (iii), not all formulas relevantly imply all classical tautologies. $\mathcal{R}$ is said to describe a more practical model for information systems because inference degrades in the presence of inconsistent propositions, unlike the classical model, where it is generally vacuous. Another advantage of $\mathcal{R}$ is that it has been shown to have better computational properties than its classical counterpart. This fact was used by Levesque to consider computational resource limitations in a model of *belief* (1984).

An interesting property about Belnap's system $\mathcal{R}$ is that one formula relevantly implies another only if they share at least one propositional variable. This property is reminiscent of 3.6.2 (vii) which states:

$$\varphi \dot{-} \lambda \vdash \varphi, \text{ whenever } \Phi_\varphi \cap \Phi_\lambda = \emptyset.$$

That is, contractions actually involve losing information only if they share at least one propositional variable. Belnap's $\mathcal{R}$ semantics is based on a 4-valued lattice (1977) regarding what has been 'told' about a proposition p. An interesting question is to what extent the semantics for $\mathcal{R}$ can be assumed by $\mathcal{T}$.

Kleene's 3-valued logic (1938) satisfies properties (ii) and (iii) but does not satisfy (i). However, the account does not consider partial orderings over the truth-values themselves. Therefore, with respect to $\mathcal{T}$, what conditions should be placed on semantic consequence to ensure *relevant* implications?

First, to say that φ *relevantly implies* λ, i.e. $\varphi \models_{\mathcal{T}} \lambda$, it should be the case that whenever φ is *true* then λ is *true*; hence, it should not be the case that φ is *true*, and λ is *false* or *no information* is available about λ. Secondly, whenever φ is *satisfiable* then λ should be *satisfiable*; hence, it should not be the case that φ is *true* or *no information* is available about φ, and λ is *false*. This can be summarized as follows:

4.5.0 Definition. For $\varphi, \lambda \in \mathcal{L}(\Phi_x)$, let

$\varphi \models_{\mathcal{T}} \lambda$ iff for all valuations $v \in V_x$,

$$\begin{aligned} C[\varphi]_x(v) = 1 &\Rightarrow C[\lambda]_x(v) = 1, \\ C[\lambda]_x(v) = 0 &\Rightarrow C[\varphi]_x(v) = 0. \end{aligned}$$

With the exception of *variable sharing*, $\models_{\mathcal{T}}$ validates every other axiom of $\mathcal{R}$. In fact, so does CPL. However, like $\mathcal{R}$, $\models_{\mathcal{T}}$ satisfies none of the classical properties indicated by (i), (ii), and (iii). Also, violations of *sharing* by $\models_{\mathcal{T}}$ are strictly governed: only if φ is classically inconsistent and λ is a tautology is sharing not satisfied:

4.5.1 Proposition. For $\varphi_0 \ldots \varphi_k$, $\psi_0 \ldots \psi_n \in \mathcal{L}(\Phi_x)$, assume $\Phi_{\varphi_0 \wedge \ldots \wedge \varphi_k} \cap \Phi_{\psi_0 \vee \ldots \vee \psi_n} = \emptyset$ and

$$\varphi_0 \wedge \ldots \wedge \varphi_k \models_{\mathcal{T}} \psi_0 \vee \ldots \vee \psi_n.$$

Then, for all *classical* $v \in V_x$,

$$C[\varphi_0 \wedge \ldots \wedge \varphi_k]_x(v) = 0 \text{ and } C[\psi_0 \vee \ldots \vee \psi_n]_x(v) = 1.$$

Hence, a large part of Belnap's $\mathcal{R}$ relevance structure is admitted through $\mathcal{T}$'s partial-order.

4.6 DATABASE SECURITY

An important topic in database theory is how to *hide* information, e.g. for security purposes. For traditional systems, this is generally accomplished by granting access to specific entity-relations expressed by the database. Queries which attempt to access an entity-relation without the correct privileges generate an exception. However, in extended database environments, the security problem is complicated by the more powerful information-handling available. In this case, the problem is that although priviledged information may not be explicitly resident within a database, various non-permitted conclusions might be obtained through inference. This is especially true of distributed environments, where each local site itself might be regarded as secure, but access to two or more sites might constitute a security breach.

There would seem to be a close connection between security issues regarding access and contraction. For example, we might *define* a database *kb* as secure with respect to φ whenever

$$kb \dot{-} \varphi \models kb.$$

Similarly, we can *generate* the database which is like *kb except* that it is secure with respect to φ. This, of course, is just

$$(kb \dot{-} \varphi).$$

Furthermore, in C at least, we can determine the local site databases which are globally secure with respect to φ. For example, for sites $kb_0 \ldots kb_n$, the 'global' database which is secure with respect to φ is given by

$$(kb_0 \wedge \ldots \wedge kb_n) \dot{-} \varphi.$$

In C, '$\wedge$' distributes over '$\dot{-}$', and so the local secure sites correspond to

$$(kb_0 \dot{-} \varphi) \ \ldots \ (kb_n \dot{-} \varphi).$$

Note that access to any sub-collection of local sites does not lead to a breach with respect to φ.

5 CONCLUSIONS

This paper has presented a model of contraction, where contracting a proposition is understood as losing information about its classical truth. The model constitutes a conservative extension of classical propositional logic (CPL). A major contribution of the approach is that non-equivalent updates are possible when contracting distinct inconsistent bases by the same proposition.

Several topics warrant further consideration; for example, looking at derivatives of the operator satisfying special properties. Also, the work on inference as 'information gain' is preliminary, and requires further study. Finally, the research has connections to conditional logic which have not been explored in this paper.

Acknowledgements

The author would like to thank, at Queen's University, Robert Tennent, and at Syracuse University, Peter O'Hearn, for extensive comments regarding earlier drafts of this work; also, to Joyce F. Wong, for proof-reading a copy of this paper.

Financial support for this research was gratefully received from the School of Graduate Studies, and the Department of Computing and Information Science, Queen's University.

References

Alchourrón, C.E., Gärdenfors, P., Makinson, D., "On the logic of theory change: partial-meet contraction and revision functions," *Journal of Symbolic Logic* (50), pp. 510-530, 1985.

Anderson, A.R., Belnap, N.D., *Entailment: The Logic of Relevance and Necessity* (1), Princeton University Press, Princeton, N.J., 1975.

Belnap, N.D., "A Useful Four-Valued Logic," *Modern Uses of Multiple-Valued Logic*, Epstein, G., Dunn J.M., eds., Reidel, pp. 8-37, 1977.

Birkoff, G., *Lattice Theory*, American Mathematical Society, Providence, R.I., 1967.

Blamey, S., "Partial Logic," *Handbook of Philosophical Logic* (III), Gabbay, D., Guenthner, F., eds., Reidel, pp. 1-70, 1986.

Borgida, A., "Language Features for Flexible Handling of Exceptions in Information Systems," *ACM Transactions on Database Systems* (10), pp. 563-603, 1985.

Dalal, M., "Updates in Propositional Databases," *Proceedings AAAI-88*, pp. 475-479, St. Paul, MN, 1988.

Fuhrman, A., "On the Modal Logic of Theory Change," *Lecture Notes in Computer Science* (465), Springer-Verlag, Berlin, pp. 259-281, 1991.

Gärdenfors, P., *Knowledge in Flux*, MIT Press, 1988.

Katsuno, H., Mendelzon, A.O., "A Unified View of Propositional Knowledge-Base Updates," *Proceedings IJCAI-89*, pp. 1413-1419, Detroit, MI, 1989.

Katsuno, H., Mendelzon, A.O., "Propositional Knowledge-Base Revision and Minimal Change," *Technical Report* KRR-TR-90-3, Department of Computer Science, Toronto, Canada, 1990.

Katsuno, H., Mendelzon, A.O., "On the Difference Between Updating a Knowledge-Base and Revising It," *Technical Report* KRR-TR-90-6, Department of Computer Science, Toronto, Canada, 1990.

Kleene, S.C., "On a Notation for Ordinal Numbers," *Journal of Symbolic Logic* (3), pp. 150-155, 1938.

Levesque, H.J., "A logic of implicit and explicit belief," *Proceedings AAAI-84*, pp. 198-202, Austin, TX, 1984.

Lownie, T.M., PhD Thesis (*in preparation*), 1992.

Satoh, K., "Nonmonotonic Reasoning by Minimal Belief Revision," *Proceedings of the International Conference on 5th Generation Computer Systems*, pp. 455-462, 1988.

Scott, D., "Continuous Lattices," *Lecture Notes in Mathematics* (274), Springer-Verlag, Berlin, pp. 97-136, 1972.

Scott, D., "Domains for Denotational Semantics," *Automata, Languages and Programming, Lecture Notes in Computer Science* (140), Springer-Verlag, Berlin, pp. 577-613, 1982.

Van Frassen, B.C., "Singular terms, truth-value gaps and free logic," *Journal of Philosophy* (63), pp. 481-495, 1966.

Weber, A., "Updating Propositional Formulas," *Proceedings of the 1st Conference on Expert Database Systems*, pp. 487-500, 1986.

Winslett, M., "Reasoning About Action Using A Possible Models Approach," *Proceedings AAAI-88*, pp. 89-93, St. Paul, MN, 1988.

A temporal revision model for reasoning about world change

M. O. Cordier
IRISA, Campus de Beaulieu
35042 Rennes Cedex, France
cordier@irisa.fr

P. Siegel
LIUP, 3 place Victor Hugo
13000 Marseille Cedex 3, France
siegel@frccup51.bitnet

Abstract

Revision can be seen as any operation that turns a cognitive state CSt into a subsequent cognitive state CSt'. Two kinds of change can be considered: in the "belief change" case, the cognitive states represent beliefs on a world; they are revised in response to the getting of new information about a static world. In the "world change" case, the cognitive states represent known facts on a real world; they are revised in response to change in this dynamic world. In the following we focus on the world change case and propose a way to keep up to date with a dynamic world. Reasoning about change requires predicting how the world will change along time. In absence of a predictive model of evolution, the commonsense law of inertia has been currently used and justifies the minimal change approach to the frame problem. We propose here to use an explicit transition model, which will be used as a predictive evolution model. Dean and Kanazawa[89] propose to use a probabilistic model of persistence and causation. We propose in this paper to use a symbolic model of transition by directly encoding expectations. In the first two sections, we describe the formalism that we propose to explicitly encode the transition model and its axiomatisation. We give then a formal definition of the revision operation using a transition model and discuss what can be a contraction operation in the context of world change. An illustrative example is presented and in the last section, our approach is compared to other related works.

1 INTRODUCTION

Revision can be seen as any operation that turns a cognitive state CSt into a subsequent cognitive state CSt'. This change is generally triggered by a "triggering event", which can be the arrival or the retraction of information. Two kinds of change can be considered: in the first case, the cognitive states represent beliefs on a world; they are revised in response to the getting of new information about a static world. This case, which we will call "belief change", has been largely studied in revision theory. In their foundational work, Alchouron, Gardenfors and Makinson [85] proposed rationality postulates for three revision operations: expansion, contraction and revision. In the second case, the cognitive states represent known facts on a real world; they are revised in response to change in this dynamic world. The first references relative to the "world change" problem were concerned with database updates [Fagin et al.83] [Winslett88a] and theory of action [Ginsberg-Smith87] [Winslett88b]. The theory of revision, worked out for "belief change", was also applied to the "world change" problem [Rao-Foo89]. It was recently stressed [Moreau] [Katsumo-Mendelzon89] that these two cases of revision have better to be distinguished. Katsumo and Mendelzon[91] argue that Gardenfors's postulates are well-suited for revision (belief change case), but propose some different postulates more adequate for updates (world change case).

In the following we focus on the world change case and propose a way to keep up to date with a dynamic world. Reasoning about change requires predicting how the world will change along time; it requires then predicting the tendency for propositions describing the world to persist along time. In absence of a predictive model of evolution, the commonsense law of inertia has been currently used. It states that a proposition remains true unless it can be proved to be false. It justifies the minimal change approach to the frame problem

[McCarthy-Hayes69]: giving a description of a world at time t, CSt, and a set of observations describing the triggering event at time t', the description of the world at time t', CSt', is the *closest* world to CSt accounting the observations. This minimal change principle is the basis of the syntax-oriented approach (PWA) proposed by [Ginsberg-Smith87] as well as the model-oriented approach (PMA) proposed by [Winslett88b] or that of Brewka and Herzberg [90].

We propose here to use an explicit transition model, which will be used as a predictive evolution model. As remarked by [Dean-Kanazawa89], given perfect knowledge of the world and a complete predictive model, it will be easy to infer the persistence of propositions. In most circumstances howewer, both are imperfectly and incompletely known and commonsense prediction has to be made under incomplete information. [Dean-Kanazawa89] propose to use a probabilistic model of persistence and causation. In this paper we propose to use a symbolic model of transition. Directly encoding expectations allow us to symbolically reason about the persistence of information along time and then to update a state CSt in its succeeding state CSt'. Non-monotonicity will result from the incompleteness of the transition model. It can be shown that using integrity (or transition) constraints, as used for database updates, [Fagin et al.83] [Cholvy86] for example, is a restricted case of our transition model and that the minimal change principle used to solve the frame problem in the theory of action [Schwind87] [Hanks-McDermott86] [Brewka-Herzberg90] can also be modelled in our framework.
It can be noticed that research on temporal issues is mainly concerned with reasoning about time as can be overviewed in [Special Issue of the International Journal of Intelligent Systems on Temporal Reasoning, vol 6, 5, 91, eds Wiley] but that very few results have been published, as far as we know, on representing how information tends to change along time.

In the first two sections, we describe the formalism that we propose to explicitly encode the transition model and its axiomatisation. We then give a formal definition of the revision operation using a transition model and discuss what can be a contraction operation in the context of world change. An illustrative example is presented and in the last section, our approach is compared to other related works.

2 FORMALISM OF THE TRANSITION MODEL

In the following, a cognitive state CSt describes the state of a world at time t. It is represented by a set of first order logical formulas closed by deduction. A formula F belonging to CSt is a formula true at time t. The revision process is triggered by a "triggering event " that conveys information about the world at time t'. This event can be the addition of a formula, corresponding to an observation, and which is asserted to be true in the succeeding state CSt'; it can also be the retraction of a formula asserted to be no longer true in the succeeding state CSt'. As usual, retracting a formula is different from adding the negation of this formula.
Revising a state CSt requires predicting how the world, and hence how its representation, will change along time. Our proposition is to have an explicit model of the transition between two succeeding states and to use it to revise a state CSt in its succeeding state CSt'. The transition model expresses then how information is supposed to change along time.

2.1 SYNTAX

The transition from a state CSt to a state CSt' is described by a set of elementary transitions on formulas. Each of these elementary transitions is expressed by a triplet < I, {Api}, F> where I, Api, and F are first order logical formulas. As classically in propositional default logic, these formulas are open formulas without quantifier; the free variables in these formulas can be interpreted as universally quantified.

- I describes the preconditions of the elementary transition on state CSt,
- F describes the effects of the elementary transition on state CSt',
- {Api} (application conditions) expresses the conditions on CSt' under which the elementary transition can take place. {Api} will be satisfied iff each formula of {Api} is consistent with CSt'($\neg$Api $\notin$ CSt'). In the following, we will note {Api} by AP; AP is said to be consistent with CSt' iff each formula of AP is consistent with CSt'.

Such a triplet can be glosed by: if I is true in a cognitive state CSt and if AP is consistent with the succeeding cognitive state CSt', then F should be true in CSt'. As it can be seen, AP gives a non monotonic aspect to this formalism and triplets can in some way be seen as temporal defaults.

This model allows us to express *constraints* (or protected formulas). A constraint F is a formula which is always true. It does not depend on time. It will be expressed by

the triplet <True, True, F>[1].

For example <True, True, Female(lulu) ∨ Male(lulu)> expresses that Female(lulu) ∨ Male(lulu) has to be true in the world at any time.

The model enables us also to express two elementary types of time-dependent formulas: persistent and remanent formulas.

A formula is said to be *persistent* if: if F is true in CSt, then it remains true in CSt'. As soon as a persistent formula has become true, it remains true along time.
A persistent formula will be described by the triplet: < F, True, F >. For example, <Dead(john), True, Dead(john)> or <Female(lulu), True, Female(lulu)> expresses that if you know at time t that "john is dead" or "lulu is a female", you will consider it true in all the succeeding states. A constraint can also be expressed as a persistent formula iff the formula belongs to the initial state.

A formula is said to be *remanent* if: if F is true in CSt, and if it is not contradictory for F to be true in CSt', then F is true in CSt'. A remanent formula will remain true as long as possible.
A remanent formula will be described by the triplet: < F, F, F >. For example, <Married(lulu), Married(lulu), Married(lulu)> or <On(cube, A), On(cube, A) , On(cube, A) > expresses that if you know at time t that "lulu is married" or "the cube is on A", you will consider it as being true as long as you cannot prove that it is false.

Also the model allows us to express other typical types of change along time such as:

< F ∧ G, Ø, F > expresses that the formula F is persistent under condition G,

< F ∧ G, F, F > expresses that the formula F is remanent under condition G,

< F, True, G > and < G, True, F> can be used, taken together, to express a "flip-flop",

< F, True, G > corresponds to a transition: if F is true in t, G will be true in t',

< F, G, G > corresponds to a default transition: if F is true in t, and if it is possible for G to be true in t', then G will be true in t' ...

Some formulas are neither constraints nor persistent nor remanent; they will be true at time t' if and only if they can be deduced from other information known to be true at time t': they are *contingent* formulas.

[1] When AP is composed of only one element, it will be noted Ap instead of {Ap} by sake of simplicity; here we note <True, True, F> instead of <True, {True}, F>.

Let us see on examples how different kinds of time-dependent information can be expressed using the transition model.

example 1:
"Someone is either single or married or divorced or widow" is a constraint; this expressed in the transition model TM by the following triplet:
<True, True, Single(x) ∨ Married(x) ∨ Divorced(x) ∨ Widow(x) >.
"A person is supposed to remain married along time, as long as the contrary is not known" expresses the remanence of the predicate Married; this corresponds to the following triplet:
< Married(x), Married(x), Married(x) >.
It is clearly the same for Divorded, Single and Widow which correspond to similar triplets.
"As soon as you are married (or divorded or widow), you cannot be anymore single" can be expressed by the persistent information:
<Married(x) ∨ Divorced(x) ∨ Widow(x), True, Married(x) ∨ Divorced(x) ∨ Widow(x) >.
"If you are married and your spouse died, you becomes a widow person" expresses a certain transition:
<Married(x) ∧ Spouse(x, y), True, Died(y) -> Widow(x)>.

example 2:
"When a watcher is in the nth room, and there is no alarm, he generally goes to the n+1th room" corresponds to a default transition:
<in-room(n), in-room(n+1), in-room(n+1)>.
"An alarm is on" is a persistent information:
<alarm, alarm, alarm>.
"When the alarm is on, the watcher generally stays in the room where he is" corresponds to a remanence under condition of the predicate in-room:
<alarm ∧ in-room(n), in-room(n), in-room(n)>.
"When the alarm is on, the watcher does stay in the room where he is" corresponds to persistence under condition of the predicate in-room:
< alarm ∧ in-room(n), True, in-room(n)>.

2.2 AXIOMATISATION

Three operators are defined on transition triplets, namely ∧ and ∨ and σ, which define an algebra on the triplets.

Let T, T1 and T2 be triplets belonging to a transition model TT.
T: < I, AP, F > , T1: < I1, AP1, F1 >, T2: < I2, AP2, F2 >
∧,∨ and σ are defined by:

$\wedge$ (T1, T2) = < I1 $\wedge$ I2, AP1 * AP2, F1 $\wedge$ F2 >
AP1 * AP2 denotes the cartesian product on the sets AP1 and AP2; if AP1 = {Api / i $\in$ I} and AP2 = {Apj / j $\in$ J}, then AP1 * AP2 = {Api $\wedge$ Apj / i $\in$ I and j $\in$ J}.

$\vee$ (T1, T2) = < I1$\vee$ I2, AP1 $\cup$ AP2, F1 $\vee$ F2 >
AP1 $\cup$ AP2 is satisfied when each formula of AP1 *and* each formula of AP2 are consistent with CSt'.

σ(T) = < σI, σAP, σF > where σ is a substitution on the free variables of T.

If TM is a given set of transition triplets, TM* is the closure of TM under these three operations.

It can be noticed that the instantiation operator and the conjunctive operator preserve the remanence and the persistence of information. *This is not the case for the disjunctive operator. The disjunction of two remanent formulas is not remanent.* Each term of the disjunction has to be consistent with the succeeding state: if On(TV, duct1) $\vee$ On(TV, duct2) is true in the state CSt, it will be true in CSt' if and only if On(TV, duct1) is consistent with CSt' and if On(TV, duct2) is also consistent with CSt'. This point is illustrated by the example 2 in IV.

examples:
Let <On(x, y), On(x, y), On(x, y)> be a triplet of TM, expressing the remanence of the predicate On for all x and y.

- By the substitution operation $\sigma_{(x = TV)}$, the triplet <On(TV, y), On(TV, y), On(TV, y)> belongs to TM*.

- From the two triplets, obtained by substitution operations, <On(TV, duct1), On(TV, duct1), On(TV, duct1)> and <On(TV, duct2), On(TV, duct2), On(TV, duct2) >, the following triplet is obtained by the $\wedge$ operation:
<On(TV, duct1) $\wedge$ On(TV, duct2), On(TV, duct1) $\wedge$ On(TV, duct2), On(TV, duct1) $\wedge$ On(TV, duct2) >.
Therefore, it also belongs to TM*.

- From the two same triplets, <On(TV, duct1), On(TV, duct1), On(TV, duct1) > and <On(TV, duct2), On(TV, duct2), On(TV, duct2) >, the following triplet is obtained by the $\vee$ operation:
< On(TV, duct1) $\vee$ On(TV, duct2), {On(TV, duct1), On(TV, duct2)}, On(TV, duct1) $\vee$ On(TV, duct2) >. It belongs then also to TM*.

3 REVISION USING THE TRANSITION MODEL

The transition model allows us to give a formal definition of the revision operation in the case of world change.

Let us first define the revision of a cognitive state CSt by the addition of an observation A. A is a first-order formula. The succeeding state CSt' is noted CSt' = Add(CSt, A).
1) The succeeding possible worlds PSt' are given by the solutions of the equation:
PSt' = Th(A $\cup$ {F / < I, {Api}, F > $\in$ TM* and I $\in$ CSt and $\forall$i $\neg$Api $\notin$ PSt'}).
2) The succeeding state CSt' is defined by:
- if the set of the possible worlds PSt' is empty, then CSt' = CSt
- otherwise CSt' = $\cap$ PSt'.
In the first case, the revision is rejected: the observation cannot be taken into account by the transition model; the observed facts are not compatible with the evolution of the world as it is given by the transition model.
In the second case, the revision is accepted and a skeptic approach is chosen: the succeeding state contains the formulas true in every succeeding possible worlds.

The case of contraction (or suppression) of an information is more subtle. The semantic of contraction in a world change context is not so clear and different interpretations can be given. It can even been argued that it has not to be considered in the world change context: a change in the world is triggered by observed facts and not by not-observed facts.

We have considered three definitions for the succeeding possible worlds when the change is triggered by the suppression of a formula A:
a) PSt' = Th({F / < I, {Api}, F> $\in$ TM* and I $\in$ CSt and $\forall$i $\neg$Api $\notin$ PSt'$\cup\neg$A})
b) PSt' = Th({F / < I, {Api}, F> $\in$ TM* and I $\in$ CSt and $\forall$i $\neg$Api $\notin$ PSt'and A $\notin$ PSt'})
c) PSt'=Th({F / <I, {Api}, F> $\in$ TM* and I $\in$ CSt and $\forall$i $\neg$Api $\notin$ PSt' and (F$\wedge$Api ->A) $\notin$PSt'}).

These three cases differ in the way they consider the requirement of not having A in CSt' versus the evolution of the world as specified by TM*. In case a) for example, the transition triplets are applied as if $\neg$A was added to CSt; this does not mean that A will not belong to CSt'. If the triplet <A, True, A> belongs to TM*, and A belongs to CSt, then A will belong to CSt'. In case b), the suppression will be rejected if it does not conform to the evolution as specified by the transition model.

Another way is to define contraction by means of the revision operation as it is done by [Katsuno-Mendelzon91]. The erasure operation as defined by [Katsuno-Mendelzon91] could be described by: CSt' = Add(CSt, True) ∨ Add(CSt, ¬A) and the symmetric erasure by: CSt' = Add(CSt, A) ∨ Add(CSt', ¬A) (where the ∨ operator denotes here an union on the models of the two resulting theories). We will use this last definition of contraction in the following.

4 AN ILLUSTRATIVE EXAMPLE

This example is the well-known example concerning Aunt Agatha's living room taken from [Winslett88b]. We want to show on this example how our formalism can be used to model the change of a world triggerred by executed actions. A restricted version will be considered: two ventilation ducts are in the living room, and three objects, a television, a newspaper and a magazine, can be moved by Tyro, the robot .

This living room is described at time t:

- by constraints which have to be satisfied by the description of the world at every time: if an object is on a duct, it is blocked; if both ducts are blocked, then the room becomes stuffy; there can be only one object at a time on a duct, an object can only be on the floor or on the ducts, ...
- and by facts which are known to be true at time t.

As we have seen before, the transition model describes how information about the world persists along time.
The set of constraints K = {Ki}, which correspond to protected formulas in [Winslett88b], are expressed as constraints by triplets <True, True, Ki> :
<True, True, Blocked(duct1) ∧ Blocked(duct2) -> Stuffy >
<True, True, On(x, duct1) -> Blocked(duct1)>
<True, True, On(x, duct2) -> Blocked(duct2)>
<True, True, On(x, y) ∧ On(x, z) -> y = z>
<True, True, On(x, duct1) ∧ On(y, duct1) -> x=y>
<True, True, On(x, duct2) ∧ On(y, duct2) -> x=y>
<True, True, On(x, y) ∧ On(z,y) -> x = z ∨ y=floor> ...
The remanence of the predicate On is expressed by the following transition triplet:
<On(x,y), On(x,y), On(x,y)>, meaning that if something is located somewhere, it is reasonable to suppose that, unless it is inconsistent, it will stay at the same place along time.

The predicates Blocked and Stuffy correspond to contingent formulas: they can be deduced from other formulas but do not persist along time from their own.

The transition model TM is then composed of the following triplets:
<True, True, Blocked(duct1) ∧ Blocked(duct2) -> Stuffy >
<True, True, On(x, duct1) -> Blocked(duct1)>
<True, True, On(x, duct2) -> Blocked(duct2)>
<True, True, On(x, y) ∧ On(x, z) -> y = z>
<True, True, On(x, duct1) ∧ On(y, duct1) -> x=y>
<True, True, On(x, duct2) ∧ On(y, duct2) -> x=y>
<True, True, On(x, y) ∧ On(z,y) -> x = z ∨ y=floor> ...
< On(x,y), On(x,y), On(x,y)>.
TM* is the closure of TM under the three operators, substitution, ∧ and ∨.

example 1.
Let us suppose that, in the initial world, the TV is on the duct1 and the magazine on the duct2. The initial description of the world CSt0 is given by the deductive closure of {On(TV, duct1), On(magazine, duct2), K}, where K are the constraints.
CSt0 = Th(On(TV, duct1), On(magazine, duct2), K).
It can be noticed that Stuffy, Blocked(duct1), Blocked(duct2), as well as On(newspaper, floor) ... are true in CSt0.

If the observation On(TV, floor) is added at time t1, (Tyro is asked to put the TV on the floor), we obtain CSt1 by Add(CSt0, On(TV, floor)):
CSt1 = Th(On(TV, floor), On(magazine, duct2) On(newspaper, floor) , K).
Following the transition model, the constraints K persist from t0 to t1. The triplet <On(TV, duct1), On(TV, duct1), On(TV, duct1)>, which results from the instantiation of <On(x,y), On(x,y), On(x,y)> cannot be applied because On(TV, duct1) is not consistent with CSt1; this is why On(TV, duct1) does not belong to CSt1. On the other hand, the triplet <On(magazine, duct2), On(magazine, duct2), On(magazine, duct2)> can be applied and On(magazine, duct2) does belong to CSt1. It is the same for On(newspaper, floor). Stuffy and Blocked(duct1) cannot be deduced anymore and thus do not belong to CSt1.

example 2: remanence of the disjunction.
Let us now suppose that the initial world is CSt0 = Th(K). The TV and the magazine are not in the room.
At time t1, Tyro is asked to put the TV on duct1 or on duct2. We get:
CSt1 = Add(CSt0, On(TV, duct1) ∨ On(TV, duct2))= Th(On(TV, duct1) ∨ On(TV, duct2), K).

At time t2, On(magazine, floor) is added. We get:
CSt2 = Th(On(magazine, floor), On(TV, duct1) ∨ On(TV, duct2), K)

The disjunctive and instantiation operators are applied on the triplets expressing the remanence of On: we obtain <On(TV, duct1) ∨ On(TV, duct2), {*On(TV, duct1), On(TV, duct2)*}, On(TV, duct1) ∨ On(TV, duct2)>. On(TV, duct1) and On(TV, duct2) are both consistent with CSt1; the disjunction persists then from CSt1 to CSt2.

At time t3, Tyro is asked to put the TV on duct1: On(TV, duct1) is added. We get
CSt3 = Th(On(magazine, floor), On(TV, duct1), K).
On(TV, duct1) ∨ On(TV, duct2) belongs to CSt3 but by deductive closure and not by remanence of the disjunction: On(TV, duct2) is no longer consistent with CSt'.

If Tyro is now asked to remove the TV from duct1, by adding ¬On(TV, duct1), we obtain at time t4:
CSt4 = Th(On(magazine, floor), ¬On(TV, duct1), K)).
On(TV, duct1) is not consistent with CSt4 and cannot persist; this is the same for On(TV, duct1) ∨ On(TV, duct2) which was true at t3; On(TV, duct1) is not consistent with CSt4 and then the disjunction cannot persist at time t4. Nothing can be said about the location of the TV except that it is not on the duct1.
If the disjunction On(TV, duct1) ∨ On(TV, duct2) was considered as having to be persistent or remanent along time, this would have to be explicitly expressed in the transition model by adding an adequate triplet. On(TV, duct1) ∨ On(TV, duct2) would then belong to CSt4 as well as On(TV, duct2) by deduction.

5 RELATED WORKS

In [Dean-Kanazawa 89], the model of persistence is represented by probabilistic information. Fluents (propositions that tend to persist whenever they become true) are distinguished from events; whether or not a fluent P will be true at time t' depends on whether or not P was true at the preceding time t. This dependency is represented using conditional probabilities p(holds(P, t')/ holds(P, t)) and p(holds(P, t')/ holds(¬P, t)). These kinds of "survivor functions" encode the change expectation of a fluent remaining true over the course of time. Knowledge of cause-and-effect relations is expressed in terms of conditional probabilities; it requires to give all possible causes for each possible effect and the probability of each effect for every possible combination of possible causes. This model of persistence can be seen as the probabilistic equivalent of our symbolic transition model.

In absence of a predictive model of evolution, the commonsense law of inertia has been currently used (a proposition remains true unless it can be proved to be false) and justifies the minimal change approaches.
[Winslett88b] used a minimal change approach, the so-called PMA as Possible Model Approach. It is possible to model her approach by specifying "protected formulas" as constraints and all other predicates as remanent. For the "Aunt Augusta" example, this corresponds to specifying On, Stuffy and Blocked as remanent predicates. No distinction is then done between the non-protected formulas in the way they persist along time, which was shown in [Winslett88b] to be not quite satisfactory. She proposed then to use prioritized predicates. The transition model seems to be a more expressive way to achieve this by directly encoding the different ways formulas persist along time: as seen in example 1, only the On formulas will be specified as remanent, which means that Stuffy and Blocked formulas will be treated as contingent formulas.
Another approach to make a distinction between remanent formulas and contingent ones is by defining the notion of maximal conformity on partial modelsas it is done in [Brewka-Herzberg90]. The revised theory does not depend on the set of formulas describing the theory as in [Ginsberg-Smith87] but depends on the set of atoms composing the partial models. The language chosen for the partial models has to be appropriately chosen in order to model the persistence of the formulas. It must correspond to what is explicitly defined as remanent formulas in our transition model.

It can be stressed that revision using the transition model is syntax-independent as it is the case with the two model-based approaches that we mentioned before: [Winslett88b] and [Brewka-Herzberg90]. Revision on two equivalent cognitive states will yield the same revised theory. This is clearly not the case with [Ginsberg-Smith 87]. As it can be seen in example 1, the fact that On(Newspaper, floor) does or does not belong to the set of formulas describing CSt0 does not modify the results; it belongs to the deductive closure and, being remanent, belongs then to CSt1.

The semantics given to the disjunctive operator by its axiomatisation is based on a possible worlds based approach of the disjunction as in [Katsumo-Mendelzon91]: it corresponds to using a case-based reasoning approach and, considering each possible world at time t, to looking for the way it changes from t to the succeeding time t'. We can see on the following classical example that we obtain results corresponding to what is intuitively wanted, as it is the case for [Winslett88b] and [Brewka-Herzberg90].
Let CSt0 be:
Th({Open(door) ∨ Open(window)});

Let us suppose that Open was define as remanent n the transition model. If we close the door, we get:
Add(¬Open(door),CSt0) =Th({¬Open(door)}).
Closing the door does not imply opening the window as it is the case with the Ginsberg-Smith approach and will be the case if Gardenfors's consistency postulate was verified.

This example is similar to the "apples and bananas" example proposed by [Morreau].

In this paper we are concerned with proposing a way to predict how information will persist along time, but an interesting point is to look to the way our model can be used to model actions. The consequences of an action can be specified by using the revision operation (Add) and the contraction operation (Sup) as it is done in the add-list and delete-list in a STRIPS-like approach. Ambiguous outcomes of actions can also be modelled as it can be seen on the following example taken from [Brewka-Herzberg90].
Tossing a coin means taking the coin which can be on head side (p) or on tail side (q) and throw it; you cannot predict what will be the actual result, except that it will be p or q. The consequences of tossing a coin can be expressed by: Add(p) ∨ Add(q). This is close to the solution proposed by [Brewka-Herzberg90] where the outcomes of this action will be given by the two partial models {p} and {q}. It should be noticed that in the case where p and q are exclusive, as it is the case here, Add(p) ∨ Add(q) corresponds to Add(p) ∨ Add(¬p) which is exactly the definition of the retraction operation given at the end of section III, i.e of the symmetric erasure as defined by [Katsuno-Mendelzon91]. Tossing a coin has as only effect to retract the information that was available about the coin.

In this paper, we focus on representing uncertain expectations on evoluting situations; we do not make comparison with works on reasoning on facts referring to time intervals (often called persistent facts) as it is done for example by [Decker88] or [Guckenbiehl91].

6 CONCLUSION

In this paper we propose to use an explicit transition model, which directly encodes expectations on the evolution of the world and will be used to update (or revise) a cognitive state when changes in the world are observed. This symbolic transition model can be compared with the probabilistic persistence model proposed by [Dean-Kanazawa 89]. A formalism for this model is defined and an axiomatisation by three operators is given. The semantics given to the disjunctive operator by this axiomatisation is based on a possible worlds approach which was shown to be adapted in a world change context [Katsumo-Mendelzon91]. It can also be noticed that the revision process does not depend on the representation used to describe the world, i.e the revision is syntax-independent. The fact that an information, true at time t, remains (or does not remain) true at the succeeding time, is explicitly expressed in the transition model.
The main approaches to the frame problem do not use an explicit model of evolution and are based on the general minimal change principle, which is simple but limited. Our formalism is powerful enough to model these approaches and proposes a more flexible way to represent and to predict the way information tends to change along time.

Acknowledgements
We would like to thank Sylvie Thiébaux and Joachim Herzberg who provided helpful comments on an earlier version of this paper.

References

[Alchourron et al 85] C. E. Alchourron, P. Gardenfors and D. Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510-530, 1985.
[Brewka-Herzberg 90] G. Brewka and J. Herzberg. How to do things with worlds: on formalizing actions and plans. Tasso-Report n° 11, 1990.
[Cholvy 86] Update Semantics under the domain closure assumption. *Proceedings of the International Conference on Database Theory*, Rome, 1986.
[Dean-Kanazawa 89]. A model for reasoning about persistence and causation. *Computational Intelligence* 5, 142-150, 1989.
[Decker 88] R. Decker, Modelling the Temporal Behaviour of Technical Systems. *Proceedings of the German Workshop on Artificial Intelligence*, pp 41-50, 1988.
[Fagin et al 83] R. Fagin, J. D. Ullman and M. Y. Vardi. On the semantics of updates in databases. *Proceedings of the second ACM conference SIGACT-SIGMOD*, pp 352-365, 1983.
[Gardenfors 88] P. Gardenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*, MIT Press, Cambridge, MA, 1988.
[Ginsberg and Smith 87] M. L. Ginsberg and D. E. Smith. Reasoning about action I: a possible worlds approach. In *Readings in Non Monotonic Reasoning*, M. L. Ginsberg and D. E. Smith eds, Morgan Kaufman, Los Altos, 1987.

[Guckenbiehl 91] T. Guckenbiehl. Formalizing and Using Persistency. *Proceedings of the IJCAI Conference*, Sydney,pp 105-110, 1991.

[Hanks-Mc Dermott 86] S. Hanks and D. Mac Dermott, Default reasoning, non-monotonic logics and the frame problem. *Proceedings of the AAAI Conference*, Philadelphia, pp 328-333, 1986.

[Katsumo-Mendelzon 89] H. Katsumo and A. O. Mendelzon. A unified view of propositional knowledge base updates. *Proceedings of the 11th IJCAI*, pp 1413-1419, 1989.

[Katsumo-Mendelzon 91] H. Katsumo and A. O. Mendelzon. On the differences between updating a Knowledge Base and Revising it. *Proceedings of KR91*, pp 387-394, 1991.

[Katsumo-Satoh 91] H. Katsumo and A. O. Mendelzon. A Unified view of Consequence Relation, Belief Revision and Conditional Logic. *Proceedings of the 12th IJCAI*, pp 406-412, 1991.

[McCarthy-Hayes 69] J. C. McCarthy and P. J. Hayes. Some Philosophical Problems from the standpoint of artificial intelligence. in B. Meltzer and D. Michie eds, *Machine Intelligence*, 4, pp 463-502, American Elsevier, New York, 1969.

[Morreau 92] M. Morreau. Planning from first Principles. in *Belief Revision*, Cambridge University Press,Gardenfors ed., 1992.

[Rao-Foo 89] A. S. Rao and N. Y. Foo. Minimal change and maximal coherence: a Basis for belief revision and reasoning about action. *Proceedings of the 11th IJCAI*, pp 966-971, 1989.

[Schwind87] C. Schwind, Action Theory and the frame problem. in *The Frame Problem in Artificial Intelligence*, F. M. Brown ed., Morgan Kaufman Publ., pp 121-134, 1987.

[Winslett 88a] M. Winslett. A Framework for Comparison of Update Semantics. *Proceedings of the 7th ACM Symposium on Principles of Database Systems*, Austin, 1988.

[Winslett 88b] M. Winslett. Reasoning about actions using a Possible Model Approach. *Proceedings of the 7th AAAI Conference*, pp 89-93, 1988.

[Special Issue on Temporal Reasoning 91], *International Journal of Intelligent Systems*, 6, 5, eds Wiley, 1991.

Computing Knowledge Base Updates

Alvaro Del Val
Robotics Lab
Stanford University
Stanford, CA 94305
delval@scottie.stanford.edu

Abstract

We present syntactic characterizations of propositional update operators based in the PMA approach [Winslett, 1988a; Winslett, 1988b]), and develop algorithms for computing the updated database on the basis of this characterization. Unlike any other previous work in this area, the methods presented here do not require direct manipulation or explicit storage of complete models of the database. They also appear to be easily extensible to other update operators based in the idea of minimal change.

1 INTRODUCTION

Much recent work has been devoted to giving accounts of belief revision based on the work of Alchourrón, Gärdenfors and Makinson ([Alchourrón *et al.*, 1985],[Gärdenfors, 1988],[Gärdenfors and Makinson, 1988]), who try to formalize the notion of a minimal belief change through a set of postulates that every revision operator should satisfy. [Katsuno and Mendelzon, 1991] recently suggested that revision is not the only way in which new information can be incorporated in a database. They propose to distinguish between revision and update, propose postulates for the latter, and provide a model theoretic characterization of update operators satisfying the postulates. Loosely speaking, revision can be seen as the operation to be performed when new knowledge about a static situation arrives, whereas updates record a change in the world. In the former, the new information might show that our initial beliefs were wrong and in need of revision, whereas in the latter our initial beliefs were correct, but the world has in the meanwhile evolved and the beliefs must be updated. More recently, [Del Val and Shoham, 1992] showed that there is a very close connection between non-monotonic theories of reasoning about action and change and a subfamily of the KM-operators, thus providing a strong foundation for the semantics of this subfamily.

In this paper we consider some operators in this subfamily, specifically Winslett's 'possible models approach' (PMA) update operators ([Winslett, 1988a; Winslett, 1988b]). We present syntactic characterizations and algorithms for the basic and prioritized version of the PMA update operators; incremental algorithms for these operators in the presence of domain constraints; and "anytime" versions of the algorithms.

Two types of results are reported in this paper, where we restrict our attention to propositional databases: first, ***syntactic characterizations*** of the result of updating an arbitrary formula with another arbitrary formula; without loss of generality, both formulas can be assumed to be in disjunctive normal form (DNF), and the methods compute the updated database also in DNF; second, ***algorithms*** based on these results. Very efficient algorithms for databases represented as a set of models or as a DNF formula can be immediately derived from them. However, storing a DNF database will often be unfeasible, so we need methods that work with more common formats, such as conjunctive normal form (CNF) or negation normal form (NNF). Databases stored in these formats are thus the most natural input ***and*** output for a realistic algorithm — in particular, the modified database must be in the same format as the original one if we want it to efficiently support other database functions such as queries. We provide algorithms for updating arbitrary CNF and NNF databases with arbitrary update formulas, returning also a CNF or NNF database. To our knowledge, these are the first results of this kind that have been reported. All previous results in this area either involve direct manipulations on complete models of the database or make essential use of having these models explicitly stored; since the set of models of a clausal database can easily be exponential in the size of its clausal representation, this is a significant limitation.

From a theoretical standpoint, there are several advantages to working with a database and update formula

in DNF. The most important is that DNF disjuncts can be seen as *partial models* of the database, which allows for manipulation of whole sets of models at once. Second, from a syntactic point of view, the update problem is reduced to the much easier problem of updating conjunctions of literals. In particular, this type of update is independent of symbols not occurring in the update formula. For clausal databases, this allows us to focus our attention in a very restricted subset of the database, a fact which is crucial for the efficiency of the algorithms presented in this paper. The algorithms presented here do incur in a very significant cost in conversion to DNF format; yet this restricted focus allows it to deal with far larger problems, it appears, than those which previously reported procedures can handle.

The structure of this paper is as follows. Section 2 and 3 contain the basic results on the syntactic characterization of PMA and prioritized PMA update operators. Section 4 is devoted to algorithms based in these characterizations. In section 5 we consider the problem of update in the presence of domain constraints or "protected formulas". Section 6 introduces anytime algorithms. In section 7 we discuss experimental results as well as some modifications to the basic algorithms that are crucial for a practical implementation. Related work is discussed in the concluding section.

In the rest of the paper, we assume a propositional language with a finite set $\mathcal{P}$ of symbols. Update operators are represented by $\diamond$, possibly subscripted. ψ always denotes the database and μ the update formula, both of which are assumed to be satisfiable. Negation normal form formulas are those obtained by closing the set of *literals* (rather than the set of propositional letters) under disjunction and conjunction. If ψ is in CNF or NNF, it consists of clauses or top level conjuncts $c_i, c_j, \ldots$. (Any NNF database can be written as $c_1 \wedge \ldots \wedge c_n$, where each c_i is a disjunction of formulas which are themselves in NNF. It's these c_i's which we will refer to as "top level conjuncts"; every CNF database is also a NNF database, with the clauses as top level conjuncts.) DNF(ψ) represents some formula in disjunctive normal form equivalent to ψ and consisting of the conjunctions of literals $\psi_i, \psi_j, \ldots$ We require that all these conjunctions be satisfiable. μ is assumed to be in DNF, and its disjuncts will be denoted by $\mu_i, \mu_j, \ldots$ Literals are represented by $l, l_i, l_j, \ldots$. With a slight abuse of notation, if there is no ambiguity a conjunction of literals will be treated interchangeably as a set of literals, and a DNF (CNF,NNF) formula as a set of disjuncts (conjuncts).

We also use *Props*(s), for a literal or set of literals s, to denote the propositional symbol or set of symbols occurring in s. *Mods*(ϕ) denotes the set of models of the formula ϕ. Finally, for any set S and ordering relation $\leq$, *Min*($S,\leq$) represents the set of $s \in S$ such that s is minimal in $\leq$ with respect to S (i.e. such that there is no $s' \in S$ with $s' \leq s$ but $s \not\leq s'$).

2 SYNTACTIC CHARACTERIZATION OF PMA UPDATE

Intuitively, PMA update operators can be seen as selecting a set of models of the update formula in terms of those which are "closer" to some model of the original database. Let *Diff*(I,J) be the set of propositional letters on which interpretations I and J differ, i.e.

$$Diff(I,J) = Props(\{p \in \mathcal{P} \mid I \models p \text{ iff } J \models \neg p\}),$$

and consider the ordering on interpretations induced by a model M by the relation:

$$I \leq_M J \text{ iff } Diff(I,M) \subseteq Diff(J,M).$$

Winslett's basic PMA update operator $\diamond$ can then be formally defined as follows:

$$Mods(\psi \diamond \mu) = \bigcup_{M \in Mods(\psi)} Min(Mods(\mu), \leq_M).$$

Example 1. Let $\psi = (b \wedge c) \vee (\neg a \wedge \neg b \wedge c)$ and $\mu = (a \wedge \neg b) \vee (\neg b \wedge \neg c)$. There are three models of ψ, namely, $M_1 = \{a, b, c\}$, $M_2 = \{\neg a, b, c\}$ and $M_3 = \{\neg a, \neg b, c\}$, and three models of μ, $N_1 = \{a, \neg b, c\}$, $N_2 = \{\neg a, \neg b, \neg c\}$ and $N_3 = \{a, \neg b, \neg c\}$. Only N_1 is minimal with respect to M_1; with respect to M_2 both N_1 and N_2 are minimal, and similarly with respect to M_3. N_3 is not minimal with respect to any of the models of ψ, and is thus ruled out. Thus, $Mods(\psi \diamond \mu) = \{N_1, N_2\}$. □

The syntactic method we are about to propose generates, for each $\psi_i \in DNF(\psi)$ and each disjunct $\mu_j \in \mu$, a DNF formula $f(\psi_i, \mu, \mu_j)$. The updated database is the disjunction of all such formulas. Notice first that this function is independent of any other disjunct ψ_j; that is, each disjunct in $DNF(\psi)$ can be updated independently[1].

If μ consisted of a single disjunct μ_j, then it is easy to see that the desired value for this function should be the formula $\delta = \bigwedge((\psi_i - Diff(\psi_i, \mu_j)) \cup \mu_j)$. For it is clear that in this case, for any model M of ψ_i the only "closest" model N of μ_j with respect to M would be the one differing from M in exactly the set of symbols (if any) which make ψ_i and μ_j inconsistent; thus N must satisfy as many literals in ψ_i as are consistent with μ_j, and thus be a model of δ. However, in the presence of other $\mu_k \in \mu$, this is no longer sufficient. N might well be minimal with respect to M in the set of models of μ_j, but not in the set of models of $\mu_j \vee \mu_k$. The way to solve this is to add to δ some

[1] This is a property shared by all KM-operators, corresponding to KM-postulate (U8): $(\psi_1 \vee \psi_2) \diamond \mu$ is equivalent to $(\psi_1 \diamond \mu) \vee (\psi_2 \diamond \mu)$.

literals so as to make it inconsistent with (some of) the other disjuncts μ_k. The completeness of the method is achieved by doing this in all possible ways. We now make this precise.

Define the difference between two conjunctions of literals ψ_i and μ_j as the set of literals in ψ_i whose negation is in μ_j, i.e

$$Diff(\psi_i,\mu_j) = \{l \in \psi_i \mid \neg l_i \in \mu_j\}.$$

For any $\mu_j \in \mu$ and $\psi_i \in \mathrm{DNF}(\psi)$, define

$$\mu^*_{\psi_i}(\mu_j) = \{\mu_k \in \mu \mid Diff(\mu_j,\psi_i) \not\subseteq \mu_k\}.$$

The idea here is that we can ignore disjuncts (in the sense that we do not need to negate any of its literals to add it to the formula computed for μ_j) whose models must differ from any model of ψ_i at least in as much as models of μ_j itself. Next, for any two $\mu_j, \mu_k \in \mu$ and any $\psi_i \in \mathrm{DNF}(\psi)$, define:

$$neg_{\psi_i}(\mu_k,\mu_j) = \{\neg l \mid l \in \mu_k - (\psi_i \cup \mu_j)\}.$$

This identifies literals in μ_k which can be profitably negated and added to the formula for μ_j. Note in particular that this will prevent us from negating any literal in $Diff(\psi_i,\mu_j)$. Finally, if $\mu^*_{\psi_i}(\mu_j) = \{\mu_1,\ldots,\mu_l\}$ then

$$neg_{\psi_i}(\mu^*(\mu_j)) = \{\mu^*_k \in neg_{\psi_i}(\mu_1,\mu_j)\times\cdots\times neg_{\psi_i}(\mu_l,\mu_j) \mid \mu^*_k \text{ is satisfiable}\},$$

where each μ^*_j is treated as a conjunction[2]. This simply makes sure that we add some complementary literal for every useful disjunct, as is made precise in the next result. (I'll drop the subscript ψ_i from now on when using any of these functions, since the context will always make clear the appropriate subscript.)

Theorem 1 *(Syntactic equivalent of PMA update.)*

$$\psi \diamond \mu \equiv \bigvee_{\substack{\mu_j \in \mu \\ \psi_i \in \mathrm{DNF}(\psi)}} f(\psi_i,\mu,\mu_j),$$

where $f(\psi_i,\mu,\mu_j) =$

$$\begin{cases} \bigwedge((\psi_i - Diff(\psi_i,\mu_j)) \cup \mu_j) & \text{if } \mu^*(\mu_j) = \emptyset \\ false & \text{if } neg(\mu^*(\mu_j)) = \emptyset \\ \bigvee_{\mu^*_k \in neg(\mu^*(\mu_j))} \bigwedge((\psi_i - Diff(\psi_i,\mu_j)) \cup \mu_j \cup \mu^*_k) & \text{otherwise} \end{cases}$$

Example 2. Let ψ and μ be as in example 1. Then

$$\begin{aligned} f(b\wedge c,\mu,a\wedge\neg b) &= a\wedge\neg b\wedge c \\ f(b\wedge c,\mu,\neg b\wedge\neg c) &= \neg a\wedge\neg b\wedge\neg c \\ f(\neg a\wedge\neg b\wedge c,\mu,a\wedge\neg b) &= a\wedge\neg b\wedge c \\ f(\neg a\wedge\neg b\wedge c,\mu,\neg b\wedge\neg c) &= \neg a\wedge\neg b\wedge\neg c \end{aligned}$$

Thus, taking the disjunction of these formulas, we obtain $\psi \diamond \mu \equiv (a\wedge\neg b\wedge c)\vee(\neg a\wedge\neg b\wedge\neg c)$, as desired. Notice that in the second line, where we are computing the formula corresponding to $\psi_1 = b\wedge c$ and $\mu_2 = \neg b\wedge\neg c$, we have that $\bigwedge((\psi_1 - Diff(\psi_1,\mu_2))\cup\mu_2) = \neg b\wedge\neg c$. Without the addition of $\{\neg a\} \in neg(\mu^*(\mu_2))$, this formula would yield an incorrect model. □

We show next that, for the special case of clausal and NNF databases, it suffices to compute the update of the subset of the database consisting of clauses or top level conjuncts sharing some propositional symbols with the update formula.

Lemma 2 *If ψ_i is a conjunction of literals, then for any $\psi_u \subseteq \{l \in \psi_i \mid Props(l) \not\subseteq Props(\mu)\}$, $\psi_{iS} = \psi_i - \psi_u$,*

$$\psi_i \diamond \mu \equiv (\psi_{iS} \diamond \mu) \wedge \psi_u.$$

This lemma tells us the result of updating a conjunction of literals ψ_i with an arbitrary formula μ depends only on the set of literals in ψ_i whose negation occurs in μ. Now we can prove:

Theorem 3 *Let ψ be a database in conjunctive (negation) normal form, with clauses (top-level conjuncts) $c_1, c_2, \ldots, c_n$. Let $\psi_S = \{c_i \in \psi \mid Props(c_i) \cap Props(\mu) \neq \emptyset\}$ be the set of top level conjuncts sharing some propositional symbols with μ, and let $\psi_U = \psi - \psi_S$. Then*

$$\psi \diamond \mu \equiv (\psi_S \diamond \mu) \wedge \psi_U.$$

It is easy to see that this theorem has a dramatic effect on the cost of computing the update. Update formulas will typically be rather short; assuming that any particular symbol occurs only in a few number of clauses or top level conjuncts, this makes the cost of the update largely independent of the total size of the database, since all other conjuncts can be ignored.

3 PRIORITIZED PMA UPDATE

We will often want to treat some facts in the world as less likely to change than others, and minimize changes in the former before minimizing changes in the latter. With this purpose, [Winslett, 1988b] defines a prioritized version of PMA update. Formally, we partition the set of symbols in the language into sets or strata $s_1,\ldots,s_n$ in descending order of priority, i.e. $p \in s_i$ has higher priority than $q \in s_j$ iff $i < j$[3]. By extension, we will say that a literal is in a stratum if its symbol

[2] Alternatively, $neg_{\psi_i}(\mu^*(\mu_j))$ can be defined as $DNF(\bigwedge_{\mu_k\in\mu^*(\mu_j)}(\bigvee neg_{\psi_i}(\mu_k,\mu_j)))$.

[3] We can also use this device for associating priorities with a formula ϕ rather than only with symbols. To do this, we introduce a new symbol p and add the protected formula $p \equiv \phi$ to the database, assigning p to the desired priority level. Protected formulas are introduced in section 5.

is in it. Let $I||_s$ represent the restriction of interpretation I to the set of symbols s. Define the difference at stratum s_k between two interpretations I and M as $Diff_k(I,M) = Diff(I||_{s_k}, M||_{s_k})$, and let

$$PDiff(I,M) = (Diff_1(I,M), \ldots, Diff_n(I,M)).$$

For any interpretations M, I, J, we say that $I \leq_M^P J$ iff for every $1 \leq k \leq n$

if $I||_{s_i} = J||_{s_i}$ for every $1 \leq i < k$, then $Diff_k(I,M) \subseteq Diff_k(J,M)$.

(As usual, $I <_M^P J$ iff $I \leq_M^P J$ but $J \not\leq_M^P I$.). Then $\diamond_P$ is defined by:

$$Mods(\psi \diamond_P \mu) = \bigcup_{M \in Mods(\psi)} Min(Mods(\mu), \leq_M^P).$$

We can obtain a syntactic characterization of prioritized PMA update by modifying the definitions in the previous section to make them sensible to the priority ordering.

For two conjunctions of literals ψ_i and μ_j define:

$$PDiff(\mu_j, \psi_i) = (Diff_1(\mu_j, \psi_i), \ldots, Diff_n(\mu_j, \psi_i))$$

We can again order the various $PDiff(\mu_j, \psi_i)$ by $<_{\psi_i}^P$, defined as: $PDiff(\mu_1, \psi_i) <_{\psi_i}^P PDiff(\mu_2, \psi_i)$ iff there exists $1 \leq h \leq n$ such that $Diff_h(\mu_1, \psi_i) \subset Diff_h(\mu_2, \psi_i)$ and for every $1 \leq j < h$, $Diff_j(\mu_1, \psi_i) \subseteq Diff_j(\mu_2, \psi_i)$. We can now redefine the function $\mu^*(\mu_j)$ (dropping, as usual, the subscript ψ_i):

$$\mu_P^*(\mu_j) = \{\mu_k \in \mu \mid PDiff(\mu_j, \psi_i) \not\leq_{\psi_i}^P PDiff(\mu_k, \psi_i)\}^4.$$

Define

$$neg_h(\mu_k, \mu_j) = \{\neg l_i \in s_h \mid l_i \in \mu_k - (\psi_i \cup \mu_j)\}.$$

(This differs from neg_{ψ_i} as defined in the previous section in the restriction to literals in a given stratum.) For each $\mu_k \in \mu_P^*(\mu_j)$, there must exist some $h \leq n$ such that $Diff_h(\psi_i, \mu_j) \not\subseteq Diff_h(\psi_i, \mu_k)$, since otherwise $PDiff(\mu_j, \psi_i) \not<_{\psi_i}^P PDiff(\mu_k, \psi_i)$ entails $PDiff(\mu_j, \psi_i) = PDiff(\mu_k, \psi_i)$. For each such μ_k, let k_{min} be the smallest integer with this property. Then

$$neg_P(\mu_k, \mu_j) = \bigcup_{1 \leq h \leq k_{min}} neg_h(\mu_k, \mu_j).$$

Finally, if $\mu_P^*(\mu_j) = \{\mu_1, \ldots, \mu_l\}$ then

$$neg(\mu_P^*(\mu_j)) = \{\mu_k^* \in neg(\mu_1, \mu_j) \times \cdots \times neg(\mu_l, \mu_j) \mid \mu_k^* \text{ is satisfiable}\}.$$

[4] We write $PDiff(\mu_j, \psi_i) \leq_{\psi_i}^P PDiff(\mu_k, \psi_i)$ iff either $PDiff(\mu_j, \psi_i) = PDiff(\mu_k, \psi_i)$ or $PDiff(\mu_j, \psi_i) <_{\psi_i}^P PDiff(\mu_j, \psi_i)$

Theorem 4 *(Syntactic equivalent of PMA prioritized update.)*

$$\psi \diamond_P \mu \equiv \bigvee_{\substack{\mu_j \in \mu \\ \psi_i \in \text{DNF}(\psi)}} f_P(\psi_i, \mu, \mu_j),$$

where $f_P(\psi_i, \mu, \mu_j) =$

$$\begin{cases} \bigwedge((\psi_i - Diff(\psi_i, \mu_j)) \cup \mu_j) & \text{if } \mu_P^*(\mu_j) = \emptyset \\ false & \text{if } neg(\mu_P^*(\mu_j)) = \emptyset \\ \bigvee_{\mu_k^* \in neg(\mu_P^*(\mu_j))} \bigwedge((\psi_i - Diff(\psi_i, \mu_j)) \cup \mu_j \cup \mu_k^*) & \text{otherwise} \end{cases}$$

Note that $f_P(\psi_i, \mu, \mu_j)$ is exactly like $f(\psi_i, \mu, \mu_j)$ in theorem 1. However, in general $\mu_P^*(\mu_j) \subseteq \mu^*(\mu_j)$ and similarly $neg(\mu_P^*(\mu_j)) \subseteq neg(\mu^*(\mu_j))$. Thus the number and size of the formulas generated in prioritized update will often be smaller than in the basic case. Again, we can restrict our attention in clausal form or NNF databases to top level conjuncts sharing propositional letters with the update formula.

Theorem 5 *Let ψ be a database in conjunctive (negation) normal form, with clauses (top-level conjuncts) $c_1, c_2, \ldots, c_n$. Let $\psi_S = \{c_i \in \psi \mid Props(c_i) \cap Props(\mu) \neq \emptyset\}$ be the set of top level conjuncts sharing some propositional symbols with μ, and let $\psi_U = \psi - \psi_S$. Then*

$$\psi \diamond_P \mu \equiv (\psi_S \diamond_P \mu) \wedge \psi_U.$$

4 ALGORITHMS

The syntactic results of the previous section can be easily transformed into algorithms for computing the updated database, both in the basic and the prioritized case. For the sake of exposition, we describe first a simple algorithm which directly translates the results in the previous sections; we consider later improvements over this basic structure.

The top level procedure simply iterates over each $\psi_i \in \text{DNF}(\psi_S)$, with the main work being done by *Update-Conj*, which computes the update for each disjunct in the DNF formula.

Procedure PMA-Update(ψ, μ)
Input: An NNF database ψ, with top conjuncts $c_i, c_j, \ldots$
A DNF update formula μ, with disjuncts $\mu_i, \mu_j, \ldots$
Output: The updated database $\psi \diamond \mu$ in CNF or NNF, where $\diamond$ represents Winslett's PMA update operator.

1. Index the disjuncts of μ by the literals occurring in each of them.
2. $\psi_S := \{c_i \in \psi \mid Props(c_i) \cap Props(\mu) \neq \emptyset\}$
3. $\psi_U := \psi - \psi_S$
4. Formulas := $\emptyset$

5. Compute one $\psi_i \in \text{DNF}(\psi_S)$
6. Formulas := Append(Formulas,Update-Conj(ψ_i, μ))
7. Go to step 5, until no more ψ_i's remain.
8. Unmark the literals of μ.
9. Return $\psi_U \wedge \bigvee$(NewFormulas).

This returns a NNF updated database. Replace $\bigvee$(NewFormulas) by CNF(NewFormulas) in line 9 if the desired output format is CNF.

Procedure Update-Conj(ψ_i, μ)
/* Output: $\psi_i \diamond \mu$ /*
1. Formulas := $\emptyset$
2. For each $\mu_j \in \mu$ do
3. InitForm := $\bigwedge((\psi_i - \mathit{Diff}(\psi_i, \mu_j)) \cup \mu_j)$
4. Diff = $\mathit{Diff}(\mu_j, \psi_i)$
5. If Diff = $\emptyset$ /* $\mu^*(\mu_j)$ must be empty too. /*
6. then Formulas := Push(InitForm,Formulas)
7. else Formulas := Append(Formulas,
Get-Final-Formulas(μ, μ_j,Diff,InitForm))
8. Return Formulas

Procedure Get-Final-Formulas(μ, μ_j,Diff,InitForm)
/* Output: The set of formulas $f(\psi_i, \mu, \mu_j)$./*
1. Neg := Get-Neg(μ, μ_j,Diff) /* $neg(\mu^*(\mu_j))$ /*
2. If Neg = NIL /* $\mu^*(\mu_j) = \emptyset$ /*
3. then return (List (InitForm))
4. else if Neg = $\emptyset$
5. then return NIL
6. else NewFormulas = $\emptyset$
7. for each $\mu_k^* \in$ Neg do:
8. Push(InitForm $\wedge \mu_k^*$,NewFormulas)
9. Return NewFormulas.

Procedure Get-Neg(μ, μ_j,Diff)
/* Output : $neg(\mu^*(\mu_j))$ /*
1. $\mu^*(\mu_j) := \{\mu_k \in \mu \mid \text{Diff} \not\subseteq \mu_k\}$
2. If $\mu^*(\mu_j) = \emptyset$
3. then return NIL
4. else NegSets := $\{neg(\mu_k, \mu_j) \mid \mu_k \in \mu^*(\mu_j)\}$
5. If Member($\emptyset$,NegSets)
6. then return $\emptyset$
7. else return DNF(NegSets)

An algorithm for prioritized PMA update can be obtained by using $PDiff(\mu_j, \psi_i)$ instead of $Diff(\mu_j, \psi_i)$ throughout, and modifying *Get-Neg* to compute $\mu_P^*(\mu_j)$ and $neg(\mu_P^*(\mu_j))$ instead of $\mu^*(\mu_j)$ and $neg(\mu^*(\mu_j))$. In the next subsection we provide details of how to do this without any increase in complexity.

4.1 COMPLEXITY

There are many possible improvements over this basic structure, but the analysis of its complexity is sufficient to bring out the major factors affecting the performance of the algorithm. We will ignore the cost of retrieving ψ_S and assume that the desired output format is NNF rather than CNF (so that we don't need to convert $\psi_S \diamond \mu$ to CNF). The worst case complexity of both algorithms when the input database is in CNF is bounded by:

$$O((\prod_{\psi_S} |c_i|)(|\mu|(\mu_{max})^{|\mu|-1})). \quad (1)$$

Here $|c_i|$ represents the size (number of literals) of the clause c_i, with the product taken over all clauses in ψ_S; μ_{max} is the maximum size of a disjunct in μ, and $|\mu|$ the number of disjuncts in μ. Since μ will typically be quite small, the crucial factor is clearly $\prod_{\psi_S} |c_i|$, which represents the worst case number of disjuncts in $DNF(\psi_S)$.

In general, the size of ψ_S will be very small relative to the total size of the database. Still, the cost is essentially exponential on the size of ψ_S, and the question is whether ψ_S will be sufficiently small in absolute terms to make this affordable. The question is thus not whether the algorithms can be used in all situations, but how large are the problems they can handle. The answer depends on whether the database is sufficiently "local", in the sense that any particular symbol will occur only in a few clauses. In that case, and given that the update formula will typically be quite small, ψ_S will also be small, and the problem appears feasible. To what extent this assumption of "locality" is satisfied in practice is an open question. At least for random databases, our experimental results suggest that rather large databases can be handled, as discussed later. Notice also that the introduction of priorities, which does not affect complexity, can have a positive effect on the feasibility of a problem.

Equation (1) should be seen only as a very rough upper bound. First, clauses in ψ_S will by definition tend to share symbols, which means that many potential disjuncts of its DNF will be either inconsistent or subsumed by others. Second, the analysis that follows will often sacrifice lower bounds for the sake of a more readable final result. The second point is mostly of theoretical interest, since it affects only the cost of *Update-Conj*. The first point, in contrast, has major practical implications.

The cost of *PMA-Update* is the sum of the costs of marking and unmarking μ (lines 1 and 8), plus the cost of computing DNF(ψ_S), plus the cost of *Update-Conj* times the number of iterations. The latter is the same as the size of DNF(ψ_S), whose worst case is $\prod_{\psi_S} |c_i|$. DNF(ψ_S) can be computed by a depth first search of the so-called "matrix" representation of a set of clauses [Bläsius and Bürckert, 1989]. In this representation, clauses are seen as ordered, and as consisting of a set of nodes (one for each literal occurrence) with outgoing edges linking each literal in a given clause to each literal in the next clause. The cost of this traversal equals the worst case size of the DNF formula, and it,

together with the cost of marking and unmarking, will disappear after simplification.

In fact, using a clever scheme of markers the traversal can be made at the same asymptotic cost while ensuring that a good number of subsumed disjuncts are not generated, that no inconsistent disjunct is generated and that, when we are done generating a given path (disjunct), the propositional letters occurring in the disjunct are all marked with the truth value corresponding to whether they ocurr positively or negatively in the disjunct. Similarly, the goal of line 1 in *PMA-Update* is to ensure that we can test in unit time whether a given symbol occurs positively or negatively in a given $\mu_j \in \mu$.

The cost for each iteration in the main loop of *PMA-Update* is the cost of *Update-Conj*, which in turn iterates over each $\mu_j \in \mu$. The cost for each μ_j is the cost of computing InitForm and Diff, plus possibly the cost of *Get-Final-Formulas*. We assume that each path is split into two sets, one containing literals with letters which occur in μ, and the other containing all other literals. Then InitForm can be computed simply by traversing the first subset (of size at most $|\mu|\mu_{max}$), collecting those literals whose symbols do not occur in μ_j; this occurrence can be checked in unit time. After collecting these literals, appending μ_j and the second set to the result gives us the desired formula. Diff can be computed in pace with this traversal, since occurrence in ψ_S can also be checked in unit time as a side effect of the depth first traversal. So the cost for this part is $O(|\mu|\mu_{max})$ per iteration.

The cost of *Get-Final-Formulas* is the cost of *Get-Neg* plus the number of formulas returned by this procedure. To compute $\mu^*(\mu_j)$, *Get-Neg* can simply traverse Diff for each μ_k; checking whether $l \in Diff$ is in μ_k is again unit time, and thus computing $\mu^*(\mu_j)$ takes a total of $|\mu|\mu_{max}$. To compute Neg-Sets, we compute $neg(\mu_k, \mu_j)$ for each $\mu_k \in \mu^*(\mu_j))$. For this, it suffices to traverse μ_k, since the literals are marked both by their presence in μ_j and by their presence in ψ_i. So NegSets takes again $O(|\mu|\mu_{max})$ time. Finally, we have the cost $(\mu_{max})^{|\mu^*(\mu_j)|} \leq (\mu_{max})^{|\mu|-1}$ of computing DNF(NegSets), for a total cost for *Get-Neg* of $O(|\mu|\mu_{max} + (\mu_{max})^{|\mu|-1})$. Since DNF(NegSets) returns at most $(\mu_{max})^{|\mu^*(\mu_j)|} \leq (\mu_{max})^{|\mu|-1}$ formulas, *Get-Final-Formulas* is thus $O(|\mu|\mu_{max} + (\mu_{max})^{|\mu|-1})$. Putting all together, *Update-Conj* takes time $O(|\mu|(|\mu|\mu_{max}+(\mu_{max})^{|\mu|-1}))$, which simplifies to $O(|\mu||\mu_{max}|^{|\mu|-1})$ for sufficiently large values. Equation (1) then immediately follows.

To show that the prioritized update algorithm has the same complexity, we assume that $PDiff(\mu_j,\psi_i)$ is computed just as $Diff(\mu_j,\psi_i)$, with its ordered structure kept implicit in the priorities of the symbols occurring in it. Then it suffices to show that $\mu^*_P(\mu_j)$ and $neg(\mu^*_P(\mu_j))$ can be computed at the same cost as its non-prioritized counterparts. For the former, traverse $Diff(\mu_j,\psi_i)$ as before, testing for each of its literals its membership in μ_k, and recording k_{min} (as defined earlier) during this traversal. Set $k = k_{min}$ if $Diff(\mu_j,\psi_i) \not\subseteq \mu_k$, or set $k = n+1$ otherwise. Then traverse μ_k. It is easy to see that $PDiff(\mu_j,\psi_i) <^P_{\psi_i} PDiff(\mu_k,\psi_i)$ iff there is some literal $l \in \mu_k$ with $l \in s_m$ for $m < k$ such that $\neg l \in \psi_i$ but $l \notin \mu_j$; for in this case (and only in this case) we have that for every $h \leq m$, $Diff_h(\mu_j,\psi_i) \subseteq Diff_h(\mu_k,\psi_i)$ and at the same time $Diff_m(\mu_j,\psi_i) \subset Diff_m(\mu_k,\psi_i)$. If no such l is found, then $PDiff(\mu_j,\psi_i) = PDiff(\mu_k,\psi_i)$ iff $k = n+1$. Thus, we can tell whether $\mu_k \in \mu^*_P(\mu_j)$ by two traversals, at a cost $O(|\mu|\mu_{max})$ to compute the whole $\mu^*_P(\mu_j)$, as it was in the case of standard update. k_{min} is computed as a side effect of this, at no extra cost, and the elements of Neg-Set can then be computed as before, except for ignoring literals in strata higher than k_{min}.

4.2 OTHER DATABASE REPRESENTATIONS

As discussed in the introduction, there are good reasons for focusing on CNF and NNF databases in the design of update algorithms. It should be obvious from the technical results presented that if the database is instead stored in DNF format in the first place then the cost of the update is *linear* in the size of the database (and exponential in the size of the update formula, but this size can be assumed to be bounded by a constant.) Alternatively, we can take a "model checking" approach ([Halpern and Vardi, 1991],[Grahne and Mendelzon, 1991]), representing the database as a set of models, each consisting of a collection of facts. We can see each of the disjuncts in a DNF formula as an specification of a partial model, and thus under the model-checking approach our approach is again linear in the size of the database.

5 PROTECTED FORMULAS

Any formula in the original database can become false as a result of PMA update. This is often undesirable. There will typically be some formulas which play the role of domain constraints, which we want to "protect" from becoming false.

Let the database $\psi = \psi_M \cup \psi_P$, where ψ_P is a set of protected formulas and ψ_M is the set of "manipulable" formulas. An update operator $\diamond_C$ under constraints can be simply defined as:

$$\psi \diamond_C \mu \equiv \psi \diamond (\mu \wedge \psi_P)^5,$$

[5]This definition, which is suggested by [Katsuno and Mendelzon, 1989] in the context of revision rather than update, is equivalent to the definition of PMA under constraints in [Winslett, 1988b], and is supported by the construction presented in [Del Val and Shoham, 1992].

i.e. we want to select interpretations in which both μ and the protected formulas ψ_P are true, choosing those in this set which differ minimally from the models of the database. (Here, $\diamond$ can be either the basic or the prioritized operator.)

Clearly, the algorithms in the previous section are inadequate for this task; in particular, the assumption that the size of the update formula is small would be violated if we used this definition directly. We present here some results which allow us to design suitable incremental algorithms with the potential to save a lot of work.

Using the fact that PMA update satisfies the KM-postulates and some results in [Katsuno and Mendelzon, 1991], we can show:

Lemma 6 *If $\psi \diamond \mu$ implies ψ_P then $\psi \diamond (\mu \wedge \psi_P) \equiv (\psi \diamond \mu) \wedge \psi_P$.*

Lemma 7 *Let $\psi = \psi_P \cup \psi_M$. For any ψ_p, if $\psi_P \models \psi_p$ and $\psi \diamond (\mu \wedge \psi_p)$ implies ψ_P, then*

$$\psi \diamond (\mu \wedge \psi_P) \equiv \psi_P \wedge (\psi \diamond (\mu \wedge \psi_p)).$$

In terms of a database ψ in clausal form, this implies that *any* subset of the clauses in ψ_P can be conjoined with μ to compute the update; the result is guaranteed to be correct whenever the resulting formula entails ψ_P.

One incremental algorithm using these facts could go as follows. Check first that the update is legal, i.e. that $\mu \wedge \psi_P$ is satisfiable. Then select some (possibly empty) $\psi_p \subseteq \psi_P$, and compute $\psi_S \diamond (\mu \wedge \psi_p)$ (where ψ_S is defined as in section 2, but in terms of $\mu \wedge \psi_p$ rather than μ alone). If $\psi_U \wedge (\psi_S \diamond (\mu \wedge \psi_p))$ entails ψ_P, then exit: this is the result of the update. Else, add some additional protected clauses to ψ_p and repeat.

Notice that the updated database $\psi \diamond (\mu \wedge \psi_p)$ entails the constraints iff for each constraint $c_i \in ((\psi_P - \psi_p) \cap \psi_S)$, the formula $(\psi \diamond (\mu \wedge \psi_p)) \wedge \neg c_i$ is unsatisfiable. Thus, we only need to check for constraints in ψ_S which are not in ψ_p, since constraints in ψ_U remain intact in the database, and those in ψ_p are guaranteed to be entailed by the modified database. (Notice though that ψ_S varies in each iteration.) Furthermore, the procedure is guaranteed to terminate with the correct update when $\psi_p = \psi_{PC} \subseteq \psi_P$, where ψ_{PC} is the set of protected clauses which are *connected* to the update formula μ, in the sense that they share propositional symbols with μ or with another connected clause.

To show this, define the set of connected clauses ψ_C and the set of connected protected clauses ψ_{PC} as follows:

$\psi_C^0 = \{c_i \in \psi \mid Props(c_i) \cap Props(\mu) \neq \emptyset\}$.
$\psi_{PC}^0 = \{c_i \in \psi_P \mid Props(c_i) \cap Props(\psi_C^0 \wedge \mu) \neq \emptyset\}$.
$\psi_C^n = \{c_i \in \psi \mid Props(c_i) \cap Props(\psi_{PC}^{n-1}) \neq \emptyset\}$.
$\psi_{PC}^n = \{c_i \in \psi_P \mid Props(c_i) \cap Props(\psi_C^n) \neq \emptyset\}$.
$\psi_C = \psi_C^n$ for any n such that $\psi_C^n = \psi_C^{n+1}$.
$\psi_{PC} = \psi_{PC}^n$ for any n such that $\psi_{PC}^n = \psi_{PC}^{n+1}$.

Assuming a finite number of clauses in ψ, both ψ_C and ψ_{PC} are well defined and finite. The following lemma tells us that ψ_C plays the same role with respect to $\mu \wedge \psi_{PC}$ as ψ_S with respect to μ.

Lemma 8 $\psi_C = \{c_i \in \psi \mid Props(c_i) \cap Props(\mu \wedge \psi_{PC}) \neq \emptyset\}$.

Theorem 9 *Let $\psi = \psi_M \cup \psi_P$ be a clausal form database with ψ_P the set of protected clauses. Let ψ_{PC} and ψ_C be as defined above, and let $\psi_U = \psi - \psi_C$. Then*

$$\psi \diamond (\mu \wedge \psi_P) \equiv \psi_P \wedge (\psi \diamond (\mu \wedge \psi_{PC})) \equiv \psi_P \wedge \psi_U \wedge (\psi_C \diamond (\mu \wedge \psi_{PC})).$$

Thus, if the proposed incremental algorithm ends up computing $\psi \diamond (\mu \wedge \psi_{PC})$, then the conjunction of ψ_P with the result is guaranteed to be the result of the update.

The procedure described here might look expensive, but checking that a database satisfies the constraints is in general intractable. An incremental algorithm can be seen as providing successive approximations which, if we are lucky, might save us a lot of work. Nevertheless, much more research is needed in this area. First, how should ψ_p be chosen at each stage so as to maximize the chance of success while minimizing cost?. Second, extensive preprocessing of constraints is likely to be needed. Finally, there is a question of whether we really need to use $\mu \wedge \psi_p$, thus effectively reinserting part of the constraints in each update, or we can instead use them as a filter for the set of formulas generated by the normal update with μ, without reinserting them. Notice that if the database is stored as a set of models, rather than as a CNF or NNF formula, then checking for entailment of constraints is polynomial, which greatly simplifies the task.

6 ANYTIME ALGORITHMS

Real-time constraints may require the interruption of the update operation before it is completed. Under these circumstances, we want to guarantee that a good approximation to the intended result can be returned at any stage of the execution of the algorithm, if it has to be interrupted, and that the approximation gets better with time. This requires the development of "anytime" versions of the algorithms, in the sense of [Dean and Boddy, 1988]. The answer can be approximated, for example, by providing an upper or lower

bound on the set of models of the updated database. Our algorithms are very well suited to providing this lower bound. The set of models determined by the algorithms as correct grows monotonically in each iteration, and no incorrect model is ever generated. Thus, the algorithms can be interrupted at any point, returning the answer accumulated so far. Notice however that since some of the disjuncts $\psi_i \in DNF(\psi_S)$ will be inconsistent with ψ_U, some of the formulas returned by the algorithms will also be inconsistent with ψ_U. Thus, if the algorithm is interrupted before processing some $\psi_i \in DNF(\psi_S)$ consistent with ψ_U, then the resulting database will be inconsistent (and the "lower bound" provided will be the empty set of models). Preprocessing techniques appear most promising in order to ensure that one such ψ_i is processed in the first iteration.

This anytime procedure applies both to standard and prioritized update. For protected formulas, we can use a similar procedure, since no matter what $\psi_p \subseteq \psi_{PC}$ we are using, we have that $(\psi \diamond (\mu \wedge \psi_p)) \wedge \psi_P$ entails $\psi \diamond (\mu \wedge \psi_P)$, and thus the accumulated result at any point of execution will be at least as strong as the final result, thus providing the desired lower bound of the set of models of the update.

Notice also that $\psi_U \wedge \mu$ provides an immediate upper bound on the models of the updated database. At this point, however, we do not know how to improve this bound other than by computing the update to completion.

7 EXPERIMENTAL RESULTS

In this section we present some preliminary experimental results about the performance of the algorithms presented in this paper. To obtain those results, however, some changes need to be made in order to obtain an efficient implementation. We describe them next.

7.1 IMPLEMENTATION

The major problem with a simple-minded implementation of the algorithms presented in this paper is that they will often generate a huge, extremely memory-hungry DNF. Many of the generated disjuncts, furthermore, will be subsumed by others, and therefore could be safely ignored. But just because of this huge size, checking for subsumption is out of the question. Rather, we need to prevent subsumed disjuncts to be generated in the first place. Furthermore, we should take advantage of every opportunity to simplify ψ_S previous to computing its DNF. After all, removing a 6-literal clause might reduce the size of the DNF by a factor of 6!

In the implementation, ψ_S was simplified after retrieval by repeatedly applying the rewriting rules

$$(l \wedge (\neg l \vee c)) \rightarrow l \wedge c;$$
$$(l \wedge (l \vee c)) \rightarrow l;$$

where l is a literal and c a disjunction of literals. Both rewriting rules preserve equivalence, and thus do not affect the result of the update. Applying these rules is essentially the same as running unit-resolution on ψ_S, and can be implemented in linear time. To maximize its effect, the simplification was performed employing all unit clauses in the database, plus the clauses in ψ_S. In addition, as a result of this procedure some clauses might be replaced by clauses which do not longer contain any occurrence of variables of μ. These clauses can be removed from ψ_S and stored back in the database. We found this procedure extremely effective in reducing the cost of computing updates.

A second procedure, somewhat more costly but also very effective, was designed to prevent the generation of subsumed disjuncts. The procedure is as follows. After applying the rewriting rules, but before beginning the depth first search, we mark every literal l_h in ψ_S which occurs in *more than one* clause $c_j \in \psi_S$ with the set $C_l(l_h)$ of clauses in which it occurs (represented as a bit-vector). Whenever one such literal is added to an evolving disjunct ψ_i, we store $C_l(l_h)$ with ψ_i. Let $C(\psi_i)$ be the set of sets of clauses thus stored with ψ_i, and let c_j be the next clause to be processed. If $c_j \in \bigcup C(\psi_i)$ then c_j can be skipped. (This is equivalent to skipping a clause c_j whenever it contains a literal which is already present in the evolving disjunct, and a more simple minded procedure would include an equivalent test.) Otherwise, before adding a literal $l_k \in c_j$ to ψ_i, we require that l_k either occurs in only one clause or both:

- $C_l(l_k) \cap \{c_1, \ldots, c_{j-1}\} \subseteq \bigcup C(\psi_i)$; and
- For every $C_l \in C(\psi_i)$, $C_l \cap \{c_1, \ldots, c_{j-1}\} \not\subseteq C_l(l_k)$.

It is not difficult to show that if l_k fails any of these two tests then there is a path (partial disjunct) through the clauses up to the current one which strictly subsumes $\psi_i \cup \{l_k\}$, and thus $\psi_i \cup \{l_k\}$ can be pruned. Experimental tests on small random databases suggest that this method eliminates a very large proportion of the subsumed disjuncts that a more simple minded procedure would generate. Again, this procedure has a major effect on the efficiency of the implementation of update. This is so because the size of ψ^S will typically allow for (bit-vector) subset tests which take a single word operation, and thus take constant time.

There are many other small improvements over this basic structure which cannot be described here in any detail. The computation of the DNF of a clausal database can be seen as a search for satisfiable paths in the "matrix representation" of clausal databases used in the connection method of theorem proving (see [Bläsius and Bürckert, 1989; Bibel, 1987;

Stickel, 1988]), which allows to import a whole set of techniques for more efficient computation of the DNF (these are not incorporated in the current implementation). There are also ways to catch in advance formulas in $f(\psi_i, \mu, \mu_j)$ which would be subsumed by other formulas also generated by the procedure. In addition, there is no need to store the complete DNF. For example, if the update formula is a conjunction, we only need to store, for each ψ_i, the set $L_i \subseteq \psi_i$ of literals which do not occur in μ. The conjunction of the update formula with the disjunction of these smaller conjuncts is guaranteed to give the result of the update. In fact, L_i would only need to be stored if it is minimal under the set inclusion with respect to the corresponding L_j's of other disjuncts. This has the potential to greatly simplify the task of converting the computed result to CNF if this is the desired output format. Slightly more complicated procedures can be used for more complicated update formulas. Preliminary experiments suggest however that the cost of checking for minimality is again too high.

7.2 RESULTS

In this section we present the experimental results of performing updates with the basic PMA operator on random databases. In designing the test suite, we made the following considerations. First, since it is the size of ψ_S rather than the total size of the database ψ that matters, we fix the size of ψ at a number high enough to demonstrate that the algorithm can work with large databases. The size of ψ_S is not chosen explicitly, since this would defeat this same purpose. Rather, we control it indirectly by varying the ratio of clauses to variables and the size of the update. We considered languages of 300, 600, 900 and 1200 symbols, and for each of them generated 50 databases with 1000 clauses. Each clause has a number of literals chosen randomly between 1 and 5 (except in the case of the 300-symbol language, where we augmented this to 6 for reasons detailed below), with each letter negated with probability 1/2. By varying the ratio of clauses to variables, we potentially affect the number of models of the database, and also the expected number of clauses in which each symbol occurs, both properties with a potential effect on the size of ψ_S. The databases were *not* checked for satisfiability, since this is not essential for testing performance, except in so far as this would make the update problem trivial. This is so when the unit resolution-like procedure described above detects the unsatisfiability of the union of ψ_S with the database unit clauses. If this happened with respect to any of the two updates with which the database was to be tested, then it was replaced by a new database. The reason for allowing 6-literal clauses in the 300-symbol language was simply to make it easier for databases to pass this test.

For each database, we randomly picked a 3-literal and a 6-literal conjunction from the given vocabulary, and computed the result of updating it with each of them, returning an NNF database. The algorithm, with the modifications described above, was implemented in Lucid Common Lisp and tested in a DEC-5000 workstation with 16Mb of memory. The results, summarized in Table 1, suggest that indeed the algorithm for basic PMA update greatly enlarges the set of feasible problems[6]. Most test problems were solved very fast, though a few of them exhausted the system memory and were interrupted before completion. Occasionally, a problem would take very long but was still solvable before our patience ran out. Including these problems in the averages would artificially inflate them. We report instead the median time for updating the 50 databases (including failures) for each choice of parameters, together with the average and standard deviation for the best 40 results. These results are reported (in seconds) in columns 4, 5 and 6 in the table; the last one indicates how many problems were interrupted before completion. Columns 1, 2 and 3 indicate the parameter values: number of symbols in the language, maximum number of literals per clause, and number of literals in the update formula.

Table 1: Experimental results: 1000-clause databases.

Sym	Max $\lvert c_i \rvert$	$\lvert\mu\rvert$	Median(50)	Best 40 Av.	Best 40 SD	Fail
300	6	3	0.008	0.008	0.003	1
600	5	3	0.012-0.016	0.017	0.019	1
900	5	3	0.008	0.012	0.011	0
1200	5	3	0.012	0.013	0.014	0
300	6	6	0.016-0.020	0.024	0.023	0
600	5	6	0.414-0.484	0.800	1.276	4
900	5	6	0.590-0.934	1.813	2.758	3
1200	5	6	0.340-0.359	1.149	2.478	4

These results clearly suggest that small updates can usually be very efficiently handled even in large databases. In the few cases in which the problem gets out of hand, the availability of anytime algorithms ensures that some useful results can be obtained. We remark that the main problem is the memory requirements of accumulating the result more than the efficiency of the procedure, which is able to generate very large results very efficiently. Storing the results in disk as it is computed may alleviate this problem, but the deeper problem is that, at least for NNF output, database growth can be very large. We have not tested yet the algorithms for CNF output. We conjecture that database growth can be greatly limited in this case, since the disjuncts generated by the procedure often share most symbols. Though the conversion from DNF to CNF is isomorphic to the opposite conversion,

[6] Retrieval time for ψ_S is not included. Rather, ψ_S was precomputed and loaded as needed, to avoid filling memory with irrelevant garbage.

the procedures for CNF to DNF conversion described in the previous subsection are only efficient when the initial set of clauses is small (for DNF to CNF, when the initial set of disjuncts is small); but since the initial set of disjuncts generated by the update algorithm is often very large, these procedures do no longer work efficiently. We remark however that from the problem solver or query processor perspective, what matters most is that a correct updated database is available for queries (and for further updates) as soon as possible. Thus, we can generate the NNF output so that the query processor can use it, and leave conversion to CNF to a later process. If, as expected, conversion to CNF greatly reduces the size of the update, then database growth would only be a temporary problem, one that we could live with in return for faster availability of the updated database.

8 DISCUSSION

In this paper we have presented syntactic characterizations and algorithms for various update operators. The characterizations are interesting in their own right, and appear to be easily adaptable to other update operators based in the notion of minimal change (e.g. operators based on cardinality counting of differences between models). We don't know of previous work of this kind in this area. As described in [Del Val, 1992], closely related characterizations are also possible for some AGM-like *revision* operators. The main difference is that for revision some of the disjuncts in the DNF database must be ignored, which can be seen as a "model pruning" operation, and has (for CNF and NNF databases) a significant negative impact in complexity.

We have also provided algorithms to compute these operators. These procedures greatly enlarge the set of feasible update problems. Previous algorithms for PMA update are all based on direct computation on complete models. [Forbus, 1989] describes, in the context of qualitative physics simulation, a procedure that appears to be an implementation of a variant of PMA update based on cardinality counting. The method is based on computing all possible states (models) during the process of 'envisionment', and it is unclear whether it can be used outside of this context. In a much more general setting, [Chou and Winslett, 1991] have recently provided an architecture for model based revision and update operators, for essentially propositional databases[7]. The details are rather tricky, but the basic idea is to compute, for each model M of the database $\psi = \psi_P \wedge \psi_M$, a set of "result models", i.e. models of $\psi_P \wedge \mu$; and then remove those result models which are not minimal with respect to M under the particular measure of distance used. The remaining (or "final") result models are exactly those in $Min(Mods(\mu), \leq_M)$. The complexity analysis they provide considers only the generation of result models from a single model of the original database. Thus, it ignores the generation of models of ψ, the possibly exponential number of such models, and the cost of testing result models for minimality. Their analysis is difficult to interpret intuitively in terms of readily ascertainable properties of the database, but is exponential in the number of literals changed in M by the update and in the height of the search tree traversed during the generation of result models. Finally, [Grahne and Mendelzon, 1991] consider the problem of PMA update of a database represented as a set of sets of "facts", each such set under the closed world assumption. They present an algorithm for non-prioritized PMA update without constraints which is linear in the size of the database thus represented. They also observed that, in the absence of protected formulas, the update of a given model is independent of symbols not occurring in μ. Since the models of the database restricted to the symbols of μ are directly available, they can simply generate models of μ and test them for minimality with respect to the database model on the letters of μ being processed. As already mentioned, our algorithms can also be applied under this "model checking" approach, and we expect them to have better performance.

Procedures based on direct manipulation of the models of the database might be adequate in situations of almost complete information, in which there are only a few models. The work of Chou and Winslett is particularly interesting in this regard; due to its extensive preprocessing of constraints, their method might well be better suited than ours to handle situations in which there is a large set of constraints but only a few models. Exploring techniques based in preprocessing of constraints in our framework is one of the topics for future research.

However, in situations of incomplete information (as most common sense reasoning problems are), the number of models can be exponential with respect to an equivalent clausal representation of the database; in these circumstances, the power of all previous methods is quite limited. This is the problem that this work begins to address. Though we did not actually compute the number of models of our test databases, it is easy to see that some of them (e.g. those in the 1200-symbol language) quite likely have a number of models which we could not even think about storing explicitly.

Other issues for further research are the extension of the techniques described here to predicate calculus, and the problem of subjunctive queries, i.e. the problem of determining whether a formula follows from the result of an update (without necessarily computing the updated database). Grahne and Mendelzon provide complexity results for subjunctive queries (see

[7] [Chou and Winslett, 1992] report some more recent results for restricted subsets of predicate calculus.

also [Eiter and Gottlob, 1991]) as well as algorithms to answer these queries under their model-checking approach. Given the close connection between PMA update and circumscription ([Winslett, 1989; Del Val and Shoham, 1992]), we are investigating the applicability of circumscriptive theorem provers (reviewed in [Brewka, 1991]) to answering subjunctive queries in clausal databases.

Acknowledgements Thanks to Yoav Shoham and Marianne Winslett for detailed comments on a previous version of this paper.

References

[Alchourrón *et al.*, 1985] Alchourrón, Carlos E.; Gärdenfors, Peter; and Makinson, David 1985. On the logic of theory change: Partial meet functions for contraction and revision. *Journal of Symbolic Logic* 50:510–530.

[Bibel, 1987] Bibel, Wolfgang 1987. *Automated Theorem Proving.* Springer-Verlag.

[Bläsius and Bürckert, 1989] Bläsius, K.H. and Bürckert, H. 1989. *Deduction Systems in Artificial Intelligence.* John Wiley and Sons.

[Brewka, 1991] Brewka, Gerhard 1991. *Non Monotonic Reasoning: Logical Foundations of Common Sense.* Cambridge University Press.

[Chou and Winslett, 1991] Chou, Timothy SC and Winslett, Marianne 1991. Inmortal: A model-based belief revision system. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning.*

[Chou and Winslett, 1992] Chou, Timothy SC and Winslett, Marianne 1992. A model-based belief revision system. Submitted.

[Dean and Boddy, 1988] Dean, Thomas and Boddy, Mark 1988. An analysis of time dependent planning. In *Proceedings of the 7th Conference of the AAAI.*

[Del Val and Shoham, 1992] Del Val, Alvaro and Shoham, Yoav 1992. Deriving properties of belief update from theories of action. In *Proceedings of the 10th Conference of the AAAI.* To appear.

[Del Val, 1992] Del Val, Alvaro 1992. *Belief Revision and Update.* Ph.D. Dissertation, Stanford University. In preparation.

[Eiter and Gottlob, 1991] Eiter, Thomas and Gottlob, Georg 1991. On the complexity of propositional knowledge base revision, updates, and counterfactuals. Technical Report CD-TR 91/23, Christian Doppler Labor für Expertensysteme, Technische Universität Wien.

[Forbus, 1989] Forbus, Kenneth D. 1989. Introducing actions into qualitative simulations. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence.*

[Gärdenfors and Makinson, 1988] Gärdenfors, Peter and Makinson, David 1988. Revisions of knowledge systems using epistemic entrechment. In *Proceedings of the 2nd Conference on Theoretical Aspects of Reasoning About Knowledge.*

[Gärdenfors, 1988] Gärdenfors, Peter 1988. *Knowledge in Flux.* The MIT Press.

[Grahne and Mendelzon, 1991] Grahne, Gösta and Mendelzon, Alberto O. 1991. Updates and subjunctive queries. Technical Report KRR-TR-91-4, Computer Science Department, University of Toronto.

[Halpern and Vardi, 1991] Halpern, Joseph and Vardi, Moshe 1991. Model checking vs. theorem proving: A manifesto. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning.*

[Katsuno and Mendelzon, 1989] Katsuno, Hirofumi and Mendelzon, Alberto O. 1989. A unified view of propositional knowledge base updates. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence.*

[Katsuno and Mendelzon, 1991] Katsuno, Hirofumi and Mendelzon, Alberto O. 1991. On the difference between updating a knowledge database and revising it. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning.*

[Stickel, 1988] Stickel, Marck E. 1988. Resolution theorem proving. *Annual Review of Computer Science* 3:285–316.

[Winslett, 1988a] Winslett, Marianne 1988a. A model-based approach to updating databases with incomplete information. *ACM Transactions on Database Systems.*

[Winslett, 1988b] Winslett, Marianne 1988b. Reasoning about action using a possible models approach. In *Proceedings of the 7th Conference of the AAAI.*

[Winslett, 1989] Winslett, Marianne 1989. Sometimes updates are circumscription. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence.*

X.

Reasoning Architectures

An Architecture for Integrating Reasoning Paradigms

James M. Skinner
Air Force Office of Scientific Research
SNL-ISRC/P.O. Box 5800
Albuquerque, NM 87185
jmskinn@cs.sandia.gov

George F. Luger
Department of Computer Science
University of New Mexico
Albuquerque, NM 87131
luger@unmvax.unm.edu

Abstract

The objective of this research is to determine what degree of synergistic behavior can be achieved from combining reasoning methodologies in a proper framework, where the strengths of one methodology compensate for the weaknesses in another, and result in a level of performance not achievable by any of the methods individually. We selected four complementary reasoning methods (case-based reasoning, rule-based reasoning, procedural reasoning, and model-based reasoning) for research. The integrating architecture modifies the traditional blackboard problem-solving model to allow multiple reasoning approaches to be combined. A control algorithm for the system is derived from heuristics for employing each of the individual reasoning methods and established blackboard control principles. A prototype demonstrates the production of a synergistic effect by diagnosing faults in a subsystem of the Hubble Space Telescope. Four aspects of the synergism are noted: cooperation, confirmation, refutation, and follow-up. We define these terms and discuss the power gain possible with a integrated reasoning approach to a problem-solving task.

1 Introduction

Earlier research [Skinner 88, Skinner & Luger 91] strongly suggested that the best approach to many problem-solving tasks may not be through a single method of reasoning, but rather by allowing several reasoning methods to be blended together. This view is compatible with that of researchers in the larger field of hybrid representation in which systems employ two or more integrated subsystems, each with distinct representation languages and inference systems. Researchers often cite the ease of expression and increase in efficiency from allowing specialized languages as the major advantages of using hybrid representation [McSkimin & Minker 79, Cohn 89, Frisch 89].

The blending of reasoning methodologies raises questions about the relationship between diverse knowledge representations and the control of an architecture for their integration. In our current research we analyze selected reasoning methodologies and use this analysis to design a system that benefits from their individual strengths while minimizing their respective weaknesses. The belief is that combining reasoning methodologies in the proper framework can result in synergism. We have designed such a framework through modifications to the traditional blackboard problem-solving model.

We developed a prototype for diagnosing faults in the Hubble Space Telescope Reaction Wheel Assembly and observed synergism through interactions best described as cooperation, confirmation, refutation, and follow-up. The remainder of this paper provides an analysis of the reasoning methods employed, a description of the prototype, an explanation of the interactions observed, and an analysis of their benefits.

2 Synergistic Reasoning

Synergistic reasoning occurs when a system employing multiple reasoning methodologies is able to solve problems that cannot be solved by any single method. To develop such a system it is necessary to select reasoning methods that are complementary, as opposed to redundant, and to design and develop a structure capable of supporting their use in an opportunistic manner. The four reasoning approaches selected for integration are: case-based reasoning (CBR), rule-based reasoning (RBR), conventional (or procedural) reasoning (CR), and model-based reasoning (MBR). We designed an architecture for integrating the reasoning methodologies by modifying the traditional blackboard architecture. We call the resulting system the Synergistic Reasoning System (SRS).

The difference between SRS and the traditional blackboard model can be understood by contrasting analogies. A common analogy used to describe the traditional blackboard model is that of a group of people trying to assemble a jigsaw puzzle on a large sticky blackboard. Each member of the group looks at his or her pieces to see whether any fit with the pieces already on the blackboard. If so, those with appropriate pieces go up to the blackboard and update the evolving solution. The new updates cause other pieces to fall into place, allowing additional pieces to be added. The entire puzzle can be solved in complete silence - there is no need for direct communication between the individuals. The apparent cooperative behavior is mediated by the state of the solution on the blackboard [Engelmore 88, Luger & Stubblefield 93].

An analogy for SRS is a person taking a closed-book test. All of the knowledge to be used during the test is self contained. However, the person is likely to use several different methods of reasoning while taking the test including relying on past experiences, employing heuristics, following procedures, or developing a mental model of a problem. These approaches roughly correspond to the machine reasoning methods of CBR, RBR, CR, and MBR respectively.

Implementing this approach requires a fundamental modification to the blackboard model. Rather than partitioning the domain knowledge functionally into knowledge sources, SRS segments the problem-solving approach into reasoning modules, with each individual module employing one of the reasoning methodologies. The system dynamically switches between the reasoning modules as necessary to solve the problem.

This approach produces a synergistic effect through *cooperation, confirmation, refutation, and follow-up*. Cooperation allows the individual reasoning modules to post partial solutions, enabling the system to solve problems that could not be solved by any single module. Thus, one module posts a partial solution not obtainable by any of the other modules, and while this module might not be able to generate the entire solution, one of the remaining modules, also unable to generate the desired solution from the original problem, is able to do so based on this new result.

Confirmation allows reasoning modules to verify results from other modules. As an example, when the RBR module recommends a tactic for solving a problem, the CBR may be able to provide past cases in which the tactic was successful. Confirmation is used to increase confidence in the conclusion or to choose between two competing tactics for problem solving.

Refutation is the ability of one reasoning module to refute conclusions of another module. That is, while incomplete information may cause one reasoning module to arrive at an incorrect conclusion, a second module may have information that disputes this conclusion. Using the same example as above, the CBR may be able to demonstrate that past attempts at solving the problem with the tactics proposed were unsuccessful. Again, this is used to increase confidence in a conclusion or to select between competing proposals.

Follow-up searches for trends in the conclusions of the system indicative of deeper problems. As an example, CBR may be used to detect repeated adjustments to a system that alleviates a problem only temporarily. This repeated occurrence of the problem is then seen as symptomatic of a deeper problem.

3 A Survey of the Selected Reasoning Methodologies

We wished to design an architecture that capitalizes on the strengths of each reasoning methodology while compensating for their individual weaknesses. As an initial step, each of the four selected reasoning methodologies were evaluated; a summary of their characteristics is shown in Table 1. A detailed analysis and explanation of the table can be found in [Skinner 92].

Obviously this is a partial list, one which will grow as research continues. The table reveals advantages unique to each individual reasoning method. Specifically, CBR employs historical knowledge and offers shortcuts, error checking, and insight. RBR employs experiential knowledge and offers speed, high performance in a limited domain, and modularity. Conventional reasoning employs procedural knowledge and offers simplicity, correctness, and verifiability. MBR employs structural knowledge and offers robustness, transferability, and causal explanations.

The disadvantages of each reasoning methodology in Table 1 can often be compensated for by one of the other reasoning modules. Cases can supplement incomplete or inconsistent rules by providing exceptions, interpretations, examples, and explanations; and can reduce the time requirements for MBR by recording results or explanations for future use. Rules can improve performance of CBR in almost all aspects of the process (i.e., bootstrapping, anticipating problems, indexing, modifying cases, verifying solutions) and can enable a MBR system to respond faster, search models more efficiently, include experiential knowledge, and focus reasoning. Models can improve CBR with causal explanations and can improve the robustness and explanation capabilities of RBR. In addition, each methodology can act as a backup in case of failure of the other methodologies.

Table 1: **Characteristics of Reasoning Methodologies (Summary).**

Method	Advantages	Disadvantages
CBR	ability to employ historical knowledge allows shortcuts in reasoning avoids past errors no domain model required existing cases for some domains knowledge acquisition relatively easy coding relatively easy clever indexing can add insight	lacks fundamental knowledge of domain complexity issues with large case base hard to define criteria for matching hard to define criteria for indexing difficult to construct/maintain index
RBR	ability to employ experiential knowledge modularity eases construction & maintenance high performance possible in limited domain simple method of providing explanations rules map naturally onto search space rules are easier to trace and debug steps in process are open to inspection separation of knowledge/control	lacks fundamental knowledge of domain cannot solve unforseen problems rapidly degrades near edges of domain explanations often inadequate knowledge is task dependent difficult to verify heuristics multiple experts may disagree
CR	ability to employ procedural knowledge correct answers when problem is constrained proven V&V techniques exist simple implementation	must have algorithm for task difficult to incorporate heuristics
MBR	ability to employ structural knowledge robust knowledge transferable between tasks can provide causal explanations versatile	lacks experiential knowledge of domain CPU/ time intensive requires an explicit domain model

4 Controlling Multiple Reasoning Paradigms

We derived a control algorithm suitable for a synergistic approach by combining principles for controlling blackboards with the findings from the survey of the reasoning methodologies. This algorithm is exercised by an Executive Module (EM) which is responsible for coordinating the problem-solving process.

In blackboard terms, scheduling knowledge sources to minimize the number of steps in a problem-solving session is known as the *focus of attention*. The developers of HEARSAY-II identified five fundamental principles for controlling the focus of attention [Hayes-Roth 77]. While these principles are defined in terms of knowledge sources, we have adapted them to the control of reasoning modules. The principles are:

(1) *The competition principle*: the best of several local alternatives should be performed first. This governs behavioral options which are locally competitive in the sense that a definite outcome of one may obviate the others.

(2) *The validity principle*: knowledge sources operating on the most valid data should be executed first. Everything else constant, the preferred knowledge source should be the one working with the most credible data.

(3) *The significance principle*: knowledge sources whose responses are most important should be executed first. This principle ensures the most important steps are performed first.

(4) *The efficiency principle*: knowledge sources which perform most reliably and inexpensively should be executed first.

(5) *The goal satisfaction principle*: knowledge sources whose responses are most likely to satisfy processing goals should be executed first.

When applied in the context of SRS, these principles led to the following set of general heuristics. From Principle (1), recommendations should be followed in order of their specificity, likelihood, and frequency. From Principle (2), recommendations should be executed according to their confidence values. From Principle (3), suspected catastrophic or time critical recommendations should be capable of preempting other tasks. From Principle (4), the most efficient reasoning modules should be used first. From Principle (5), top-level goals should have priority over sub-goals.

Additional heuristics derived from the advantages of each methodology as given in Table 1 suggest that:

(1) CR should be used whenever a polynomial-time algorithm exists; (2) CBR should be used if no specific recommendation is present, and as a means of error checking; (3) each of the reasoners should be used as a failure backup to the others; (4) RBR should be used for quick fixes, if no causal explanation is required, or if time constraints are strict; (5) MBR should be used if a causal explanation is required, but only if adequate time is available; and (6) the results of the sessions should be stored by the CBR module.

The resulting guidelines for a diagnostic application are shown below grouped by the principle from which they were derived. These guidelines are by no means static - the intent is for the set to grow and to be refined as dictated by results of continuing research in integrating reasoning paradigms. While no priority is intended, we chose to implement them in the order of appearance.

From the efficiency principle:
(a) If two reasoning modules can perform a task,
    choose the most efficient for that task.
(b) If status is nominal,
    employ the CR module.
(c) If more than one reasoner can act on a goal,
    employ in the order of RBR, CBR, MBR.
(d) If no specific recommendation is present,
    employ the CBR module.
(e) If no specific recommendation exists & CBR fails,
    employ the MBR module.
(f) If no recommendation exists & CBR, MBR fail,
    employ the RBR module.

From the competition principle:
(g) If multiple components are suspected,
    diagnose the most specific (lowest level).
(h) If multiple recommendations exist,
    perform the one closest to isolating a fault.
(i) If multiple components are suspected,
    diagnose the least reliable.
(j) If multiple components are suspected,
    diagnose the one with the most recommendations.

From the significance principle:
(k) If a symptom could be catastrophic,
    diagnose that symptom first.
(l) If a recommendation is time critical,
    perform it first.

From the strengths of the individual methodologics:
(m) If time is constrained,
    employ RBR.
(n) If no causal explanation is required,
    employ RBR.
(o) If causal explanation required & time allows,
    employ MBR.
(p) If the diagnosis session is complete,
    employ CBR to store session results.

From the goal satisfaction principle:
(q) If multiple goals exist,
    act on the top-level goals first.

From results of current research:
(r) If a fault has been diagnosed,
    initiate confirmation, refutation, & follow-up.

Guideline (a) states the most efficient and reliable reasoning module will be used to solve a problem; this is the general case for all reasoning modules. Guidelines (b)-(f) implement (a) for ordering the recommendations of the four selected reasoning modules and selecting between competing modules to achieve goals. Guideline (b) is the specific case in which no fault has occurred. In this case, conventional algorithms exist capable of handling the situation at a lower cost (in terms of time and space requirements) and with a higher reliability than any other reasoning method.

Guideline (c) presents the criteria for choosing between competing reasoning modules to perform a task. The priority used is to rely on RBR, then CBR, then MBR. In general, robustness increases and efficiency decreases in order of CBR, RBR, and MBR. By favoring RBR, a balance between robustness and efficiency is achieved. CBR is selected second because it can be executed quickly.

Guidelines (d), (e), and (f) handle the situation when no specific recommendations are present. Under these circumstances, the reasoning modules are prioritized as CBR first, then MBR, then RBR. The Executive Module forms a goal for the CBR to match on the list of symptoms in the current session; the CBR module returns a recommendation to diagnose the faulty component from a similar past case. Next, a goal is set for the MBR to diagnose each of the components in the list of suspected components (if the list is empty, a goal is created to diagnose the model of the entire system). Finally, a goal is created for the RBR module to diagnose the current list of symptoms and return appropriate recommendations. The rationale behind the prioritization is that the information available favors CBR over MBR, and MBR over RBR. At least one symptom is guaranteed to be present (otherwise the CR module would be in control), and the case base is indexed by symptoms. The list of suspects provides a focus for the MBR to diagnose the fault. While the RBR may be able to diagnose the symptom, it was unable to do so with the information available at the time the symptom was first recorded.

Guidelines (g) and (h) are implemented by tracking the level of the subcomponent suspected. The entire system under diagnosis is designated Level 1 with all direct subcomponents assigned to Level 2, and in general, subcomponents of a component on Level n are assigned Level n+1 . Under this scheme, a fault isolated to Level n is at a lower level and more specific than a fault isolated to Level n-1. For purposes of this

research, a heuristic is employed that faults isolated to a lower level are closer to isolating the fault; this is strictly true only if all subcomponents have the same number of levels.

Guideline (i) advises that the least reliable of the components suspected be diagnosed first because this is the component most likely to be the cause of the fault. This is implemented by comparing the values of the Reliability slots of the components. Reliability is expressed as the mean time between failure, in hours, for the component.

Guideline (j) favors the most frequently proposed recommendation. It is implemented by counting the number of occurrences pending for each recommendation and executing the recommendation with the highest number of occurrences.

Guidelines (k) and (l) are from the significance principle; catastrophic or time critical events should be handled first. Guideline (k) is implemented by diagnosing catastrophic symptoms first. Guideline (l) is implemented by considering time-critical recommendations first. Both catastrophic symptoms and time-critical recommendations are application dependent and determined *a priori*.

Guidelines (m)-(o) arise from the strengths of the reasoning methodologies and require knowledge of the user's desires. They are implemented through messages posted on the blackboard. That is, a user may post that a causal explanation is required or that time is constrained on the blackboard. The default values are that a causal explanation is not required and time is not constrained.

Guideline (p) dictates the problem-solving information be stored for use in future diagnostic sessions. This is accomplished by the CBR module.

Guideline (q) is due to the goal satisfaction principle. Goals in the system are stored hierarchically; a top-level goal may have sub-goals. This guideline is implemented by acting on top-level goals first (i.e., goals without links to higher level goals).

Guideline (r) is a result of this research and the discovery of how SRS can produce synergism. It is implemented by posting the three goals when a diagnosis is reached. Each module then responds according to its ability.

5 The SRS Prototype

We constructed a prototype of the Synergistic Reasoning System for diagnosing faults in the Hubble Space Telescope (HST) Reaction Wheel Assembly (RWA). The function of the HST RWA is to point the Space Telescope at the proper area of the sky and keep the telescope locked onto its target. The RWA functions according to the principle of conservation of angular momentum. When the telescope is stationary, the reaction wheel moves at a small speed to counteract the torque caused by Earth's gravitational field. To move the telescope, the speed of the reaction wheel is increased, causing the telescope to spin in the opposite direction. When the telescope nears its proper orientation, the spin is reversed and the telescope slows down. There are four reaction wheels aboard HST, and the sum of the torque forces generated by these wheels enables the telescope to rotate about an arbitrary axis [Keller 90].

5.1 Structure for a SRS Prototype

The SRS prototype was implemented in a commercial shell known as the Generic Blackboard (GBB), a toolkit based on the Common LISP Object System (CLOS) [BBT 91]. GBB provides the facilities required to construct a typical blackboard application including the blackboard database, knowledge sources, and the control shell.

SRS is a modified blackboard architecture with a hierarchical blackboard database, four reasoning modules (i.e., CBR, RBR, CR, and MBR), and an Executive (control) Module. The blackboard database has one root blackboard and four blackboards as interior nodes (one each for the individual reasoning modules). The root blackboard has seven spaces: Status, Symptoms, Suspects, Actions, Diagnosis, Recommendations, and Goals. The interior blackboards each have a single space to record local information.

The CBR module is implemented through GBB's pattern-matching facilities. We currently maintain a case base that includes a case number, the source of the case (either actual or hypothetical), the list of symptoms, the suspected components, the diagnosis, the list of actions taken to correct the fault, and the result (success or failure). The identification number comes from a LISP function call to universal time; the case number therefore serves not only as a unique identification number, but a means by which the CBR module can employ temporal reasoning during follow-up.

The RBR module uses the embedded GBB/OPS inference engine as a means of implementing a rule-based system. The RBR module treats the symptoms and goals of the problem-solving session as facts, asserting them into its knowledge base. It begins a data-driven inference resulting in the creation of recommendations or actions to be taken. As it fires each rule, the RBR module records its consequence on the RBR blackboard.

While the CR module can be involved in the diagnosis process, its primary purpose is to reason about the domain in the absence of any faults. During normal operation, the CR module posts messages from the user on the blackboard concerning expected out-

ages or anomalies. When a fault is detected, the CR module posts the symptoms on the blackboard and surrenders control. The CR module for our prototype is implemented in CLOS.

The MBR module diagnoses the suspected components to determine the likely cause of the symptoms. The MBR module is implemented in CLOS and uses the principle of locality. This principle considers how components are connected (mechanically, electrically, physically) to determine how behavior of one component can be influenced by another component [Davis 85].

The Executive Module (EM) exercises explicit control over SRS by determining the order in which the reasoning modules work on the problem and coordinating the problem-solving process. It is implemented through a combination of GBB's control shell, knowledge sources, and CLOS.

5.2 Cases for the Domain

Cases were constructed based on consultations with a satellite analyst [Campbell 92]. A sample case is shown below. The case number is universal time, representing the number of seconds since midnight, January 1, 1900 GMT. The case number will be used as a unique identification number and a means by which the CBR module can reason temporally during the follow-up phase, searching recent actual cases for trends in diagnosis.

```
Case-Number: 2902248000 ;;; Dec 20 1991 1500
Symptoms: ((:weak-signal)
           (:calibrate-pointing :unsuccessful))
Suspect: none
Actions: ((:cr :symptom-posted :weak-signal)
          (:em :check-prior-messages :none)
          (:rbr :adjust-antenna :unsuccessful)
          (:cr :calibrate-pointing :unsuccessful)
          (:mbr :diagnose-acs :acs-faulty))
Diagnosis: attitude-control-system
Result: successful
Source: actual
```

5.3 Rules for the Domain

The rules are a set of diagnostic associations relating the readings of the temperature sensors to the possibility of faults in the bearings or electronics. An example of one such rule concerning the rotor control electronics (RCE) is:

IF Temperature of RCE-Bearing-Sensor is High,
and Temperature of RCE-Sensor is OK,
and Temperature of Tunnel-Sensor is OK
THEN Set Malfunction of RCE-Bearing to True.

This rule states that if the sensor for RCE-bearing is abnormally high, and nearby sensor readings are normal, then there must be a malfunction within the RCE-Bearing [Keller 90].

5.4 Procedures for the Domain

As noted in the section on control guidelines, the conventional reasoner is responsible for reasoning about the environment as long as the status of the system is nominal. For the HST RWA, the knowledge required is the set of control algorithms that are currently used onboard the vehicle. For purposes of the prototype, the CR module implements a simulator for the attitude control system that enables the satellite to maintain its correct position and attitude. The simulator allows the user to change the attitude of the satellite relative to the Earth or to change the path of the satellite around the Earth. The simulator fires the thrusters as necessary to achieve the new position and reflects the changes through graphics on the screen.

The CR module also acts as the interface between the user and SRS. It allows the user to post messages concerning scheduled maintenance on the blackboard and to induce a fault in any of the components of the HST. The CR module provides access to the GBB graphics facilities which allow the user to view the objects posted on the blackboard, examine their slots, and follow links from one object to another.

5.5 Models for the Domain

The primary knowledge source for the models used in the prototype of the SRS was a set of papers written by researchers from NASA Ames and Stanford [Keller 90, Gruber 90] that cover the structural and functional models for the HST RWA. Secondary sources were used to provide details for constructing models. The structure and function of additional components on which the RWA depends were taken from a satellite design manual [Wertz 91]. In addition, we held knowledge engineering sessions with a satellite operator [Garnham 90] and a satellite analyst [Campbell 92] to determine how failures in the system may reveal themselves as symptoms. The resulting set of models is a composition of the knowledge from these sources.

6 Sample Operation of SRS

We developed a scenario to test the operation of SRS in which the onboard sensors detected a weak signal from the ground station. The response of the system is useful in depicting the four aspects of synergistic behavior. The state of the blackboard at various points is shown in the figures. An explanation of the events that led to these states follows.

6.1 Diagnosis of the RWA

The scenario begins during normal operations, with the CR module active. When the signal strength falls below a predetermined level, the CR module posts the symptom on the blackboard and surrenders control. The creation of a symptom causes the status to change to a fault condition which, in turn, triggers the EM.

After determining the symptom is not due to scheduled maintenance, the EM posts a goal to diagnose the symptom. The RBR responds using a set of rules that it has concerning the antenna adjustment which allow it to increase the gain by ten percent or to calibrate pointing. The RBR recommends an increase in gain which boosts the signal and alleviates the problem. The diagnosis is low-gain and (it would seem) the diagnostic session is complete. The posting of the diagnosis causes the EM to add three goals: confirm the diagnosis, refute the diagnosis, and follow-up on the diagnosis. This is SRS's method for error checking and increasing confidence in the conclusion. The state of the blackboard at this point is shown in Figure 1.

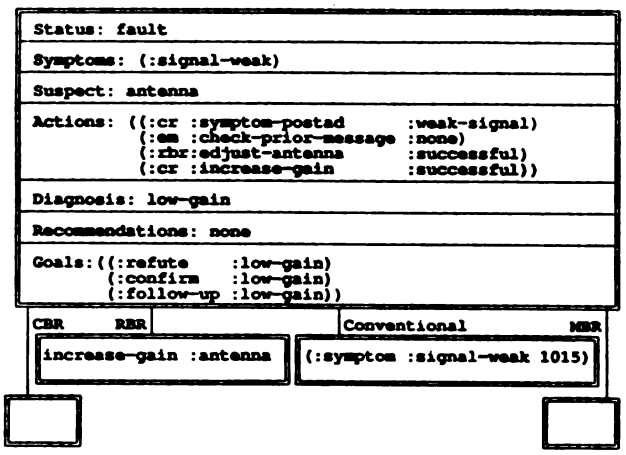


Figure 1: **Sample Operation.** The upper portion of the figure shows the contents of the seven major spaces of the top-level blackboard. The lower four boxes reveals the contents of the individual modules.

The CBR confirms the actions taken are the correct response for the given symptom by finding a past case which resulted in success. Next, the CBR attempts refutation, but cannot find any cases in which this tactic was unsuccessful. During follow-up, the CBR discovers that the gain has been increased twice in the last three hours. This trend, seen as indicative of a deeper problem, is posted as a new symptom and diagnosis is continued.

The RBR recommends calibrating the pointing of the antenna, but the CR module reports that calibration failed. This is added as a new symptom and causes the

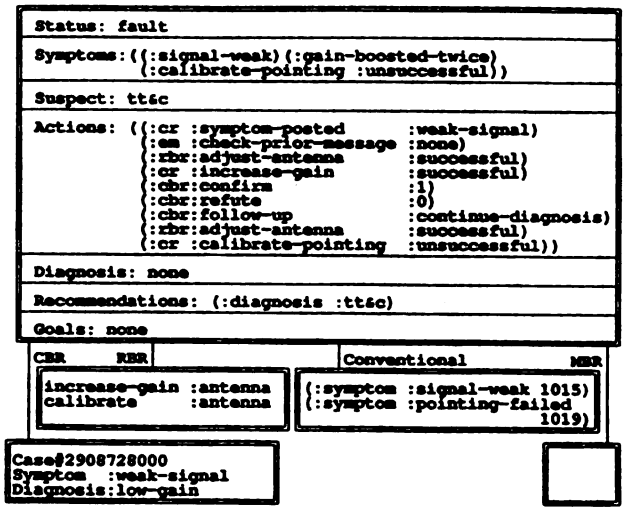


Figure 2: **Sample Operation (cont'd).** Through *follow-up* and *cooperation* additional symptoms have been identified. The CR module has recommended that the TT&C subsystem be diagnosed.

Tracking, Telemetry, and Control (TT&C) subsystem to be suspected. The state of the blackboard at this point is shown in Figure 2.

The MBR module constructs a model of the TT&C subsystem and checks each point, but no fault is found. At this point, the EM has no specific recommendations and must rely on the predetermined guidelines. The CBR module is used to search for past cases, retrieving a case in which the antenna could not be calibrated due to a fault in the attitude control system (ACS).

The MBR builds a model of the ACS and isolates the fault to the RWA. It cannot, however, find any malfunction in the components of the RWA model. The RBR module uses experiential knowledge to determine the faulty behavior is due to a high ambient temperature in the bay and recommends opening a louver to the outside to allow heat to dissipate; closed louver is posted as the diagnosis. Once again, confirmation, refutation, and follow-up are posted as goals.

No confirmation is found, but the CR refutes the diagnosis - according to its data the louver is open. The EM relies on the MBR to resolve the contradiction. In diagnosing a model of the thermal control system (which contains the louver), the MBR determines the input to the louver motor is good, but the louver is closed. The motor is determined to be bad and a backup motor is employed. The final state of the blackboard is shown in Figure 3. The results are stored by the CBR module, the blackboard is scrubbed, and control is returned to the CR module.

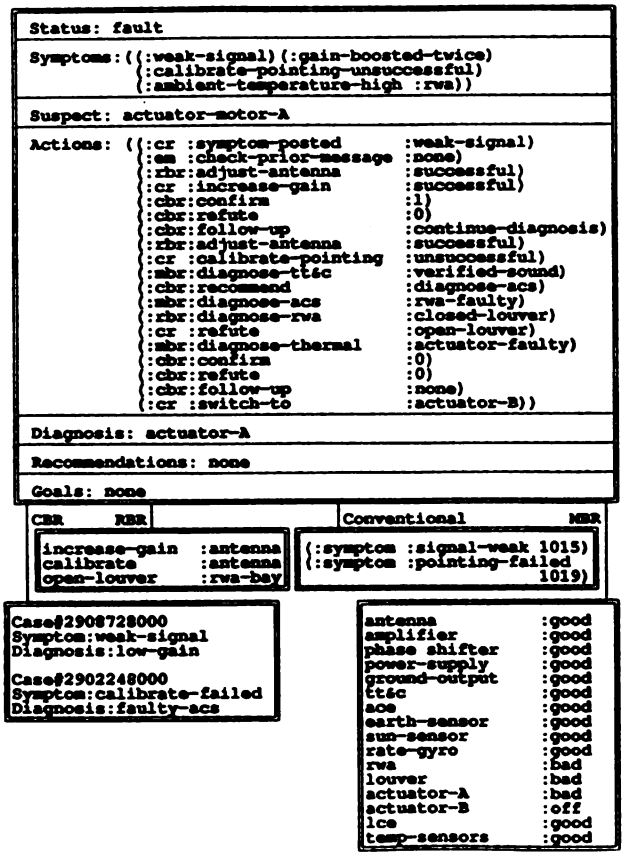


Figure 3: **Sample Operation Final State.** The fault has been isolated to the actuator motor. The results will be stored by the CBR module for use in future sessions.

6.2 Four Aspects of Synergism

This example illustrates the four aspects of synergistic behavior: cooperation, confirmation, refutation, and follow-up. While the original symptom was a weak signal, the actual cause was the failure of an actuator motor in the thermal control system. This failure caused the louver to remain closed, thereby overheating the reaction wheel assembly. This in turn prevented the attitude control system from maintaining the correct attitude, causing the antenna to be improperly calibrated. As a result, the signal strength continually degraded, and a weak signal was observed.

Individually, none of the reasoners would have responded with a correct diagnosis. Both the CBR and RBR modules would have attributed a weak signal to low-gain. The CR module had no algorithm to solve the problem. The MBR module would have diagnosed the TT&C model, only to find all components were sound. Yet, the reasoning modules were able to produce a proper response by collaborating on the problem.

Cooperation is the ability to construct a solution from partial postings. Cooperation was apparent as the reasoning modules worked together to isolate the problem. The CBR used historical knowledge to determine the inability to calibrate the antenna could be due to a fault in the ACS. The MBR used structural knowledge of the ACS to isolate the problem to the RWA, but, since heat flow was not included in the model, was unable to determine the cause of the faulty behavior. The RBR used experiential knowledge to identify the source of the problem as a closed louver.

Confirmation, the ability of one reasoning module to verify the results of another module, was demonstrated by the use of the CBR module to increase the confidence of the decision to increase the gain of the antenna. While this was only a temporary fix, it was the correct response for the available information.

Refutation was exercised when the CR reported that the louver was already open. The RBR module had incomplete knowledge of the current configuration of the system, leading to an erroneous conclusion that the louver was closed. The additional information provided by the CR led to mediation of the contradiction by the MBR.

Finally, follow-up is the ability to identify trends indicative of deeper problems. This aspect of synergistic behavior occurred when the CBR noted the repeated gain increase. Had this not been noted as a symptom of a deeper problem, an autonomous system might have continued to increase the gain, without addressing the underlying thermal problem which could eventually cause permanent damage.

7 Conclusions

We have designed an architecture that allows diverse reasoning paradigms to be integrated in a cohesive manner. We have enhanced the advantage of this integration by selecting four reasoning methods that are complementary in that they provide a convenient manner to gather and represent contrasting knowledge. During the knowledge engineering phase of development the use of multiple approaches allows the problem to be viewed from many angles, resulting in a more complete picture of the domain. During execution, the system employs this diverse knowledge in a collaborative fashion to capitalize on the collective advantages of the methods shown in Table 1, while diminishing the effect of their individual weaknesses.

The combination of the paradigms provides an ability to employ historical, experiential, procedural, causal, and structural knowledge during a problem-solving session and thus enables SRS to solve all problems solvable by any of the four reasoning methodologies individually. The control guidelines developed from established principles of blackboard control and our

research of reasoning characteristics allow the system to produce a synergistic effect through cooperation, confirmation, refutation, and follow-up. The prototype demonstrated this synergistic effect by solving a problem that none of the individual reasoning methodologies could solve.

While we have presented SRS as an approach to diagnostics, it represents an efficient and robust problem-solving model that can be applied to any domain suitable for one or more of the four reasoning methodologies employed. It also provides a basis for future research in integrating reasoning paradigms.

References

Blackboard Technology Group. *GBB User's Guide Version 2.0.* Amherst, MA: BBT Group, February 1991.

Campbell, William S., Satellite Analyst. Personal interviews. Kirtland AFB, NM, March 1992.

Cohn, A. G. "On the Appearance of Sortal Literals: a Non Substitutional Framework for Hybrid Reasoning," *Proceedings of the First International Conference on Knowledge Representation*, edited by R. Brachman et al , 55-66. San Mateo, CA: Morgan Kaufmann (1989).

Davis, Randall. "Diagnosis Via Causal Reasoning: Paths of Interaction and The Locality Principle," *Artificial Intelligence in Maintenance*, edited by J. Jeffrey Richardson, 102-122. Park Ridge, NJ: Noyes Publications, 1985.

Engelmore, R.S. and A.J. Morgan. *Blackboard Systems.* Wokingham, England: Addison-Wesley Publishing Company, Inc., 1988.

Frisch, Alan M. "A General Framework for Sorted Deduction: Fundamental Results on Hybrid Reasoning," *Proceedings of the First International Conference on Knowledge Representation*, edited by R. Brachman et al, 126-136. San Mateo, CA: Morgan Kaufmann (1989).

Garnham, John, Satellite Operator. Personal interviews. Kirtland AFB NM, October 1990.

Gruber, Thomas and Yumi Iwasaki. "How Things Work: Knowledge-based Modeling of Physical Devices," KSL 90-51. Stanford, CA: Stanford University, August 1990.

Hayes-Roth, Fredrick and Victor R. Lesser. "Focus of Attention in the Hearsay-III System," *Proceedings of the International Joint Conference on Artificial Intelligence*: 27-35 (August 1977).

Keller, Richard et al. "Model Compilation: An Approach to Automated Model Derivation," NASA report RIA-90-04-06-1, April 1990.

Koton, Phyllis. "Reasoning about Evidence in Causal Explanations," *Proceedings of AAAI-88*: 256-261 (1988).

Luger, George F. and William A. Stubblefield. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* Redwood City, CA: Benjamin/Cummings, 1993.

McSkimin, James R. and Jack Minker. "A Predicate Calculus Based Semantic Network for Deductive Searching," *Associative Networks: Representation and Use of Knowledge by Computers*, edited by Nicholas Findler, 205-238. New York: Academic Press, 1979.

Skinner, James M. *A Diagnostic System Blending Deep and Shallow Reasoning*, MSCE Thesis, AFIT/GCE/ENG/88D-5, Air Force Institute of Technology, OH, December 1988.

Skinner, James M. and George F. Luger. "A Synergistic Approach to Reasoning for Autonomous Satellites," *Proceedings of the Advisory Group for Aerospace Research and Development (AGARD) Panel on Machine Intelligence for Aerospace Electronic Systems* (1991).

Skinner, James M. *A Synergistic Approach to Reasoning*, Ph.D. Dissertation, Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, May, 1992.

Wertz, James R. and Wiley J. Larson. *Space Mission Analysis and Design.* Dordrecht, Netherlands: Kluwer Academic Publishers, 1991.

Concurrency Control for Knowledge Bases

Vinay K. Chaudhri **Vassos Hadzilacos** **John Mylopoulos**
Department of Computer Science, University of Toronto
Toronto, M5S 1A4, Ontario, CANADA

Abstract

As the demand for ever-larger knowledge bases grows, knowledge base management techniques assume paramount importance. In this paper we show that large, multi-user knowledge bases need concurrency control. We discuss known techniques from database concurrency control and explain their inadequacies in the context of knowledge bases. We offer a concurrency control algorithm, called the Dynamic Directed Graph (DDG) policy that addresses the specific needs of knowledge bases. The DDG policy exploits the rich structure of a knowledge base to support the interleaved, concurrent execution of several user requests, thereby improving overall system performance. We give a proof of correctness of the proposed concurrency control algorithm and an analysis of its properties. We demonstrate that these results from concurrency control interact in interesting ways with knowledge base features and highlight the importance of performance-oriented tradeoffs in the design of knowledge-based systems.

1 INTRODUCTION

Very large knowledge-based systems will soon be commonly upon us. With this, issues that have occupied the database world will come to concern KR developers, although perhaps complicated in interesting ways by the logical interpretation of KR languages.
– Ron Brachman, AAAI-90 Invited Lecture.

As we build ever larger knowledge bases, it is reasonable to expect that the time will soon arrive when it will no longer be viable or desirable to maintain multiple copies of the same knowledge base for each one of its users, nor will it be economically feasible to restrict access to the knowledge base to one user at a time. Instead, it is expected that multiple users will share a single knowledge base which receives queries and updates and interleaves their execution against the knowledge base, thereby optimizing the deployment of computing resources, both CPU cycles and space.

To make the problem more concrete, consider the construction of a large knowledge base, part of which is stored in the primary storage and the rest in the secondary storage[1]. If query/update requests from users are processed sequentially, the system will remain idle while waiting for a disk access to complete (see Figure 1(a): dark dots and vertical lines correspond to the requests of two different users). However, if requests are processed in an interleaved fashion, i.e., concurrently as suggested in Figure 1(b), the idle periods can be reduced resulting in a higher system throughput[2] and a quicker response time to the user (see Figure 2). This improvement in throughput or response time will grow with the number of users until resources available to the system become saturated[3]. Beyond this point, interleaved execution of user requests does nothing to enhance the system performance and may cause it to deteriorate because of the overhead of concurrency control.

In addition to improving the performance of a system, concurrency promotes sharing of knowledge, which is expected to be increasingly important in the future (Neches et al. 1991).

In the present paper, we concentrate on the problem of concurrency control. We propose a concurrency control algorithm that suits the requirements of knowl-

[1] It will not be possible to fit a large knowledge base such as CYC (Lenat and Guha 1989) in main memory.

[2] Measured by the number of user requests executed on average per unit of time.

[3] Concurrent processing can lead to arbitrary improvements in performance — sometimes of the order of *one hundred times* (Gray 1992). Even when there are no disk waits, performance studies indicate that a round robin scheduling discipline that allows interleaved processing of user requests gives best overall performance (Lazowska et al. 1984).

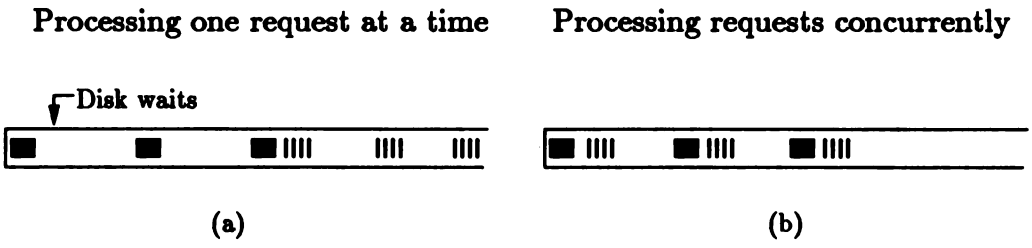


Figure 1: Concurrent processing reduces idle times

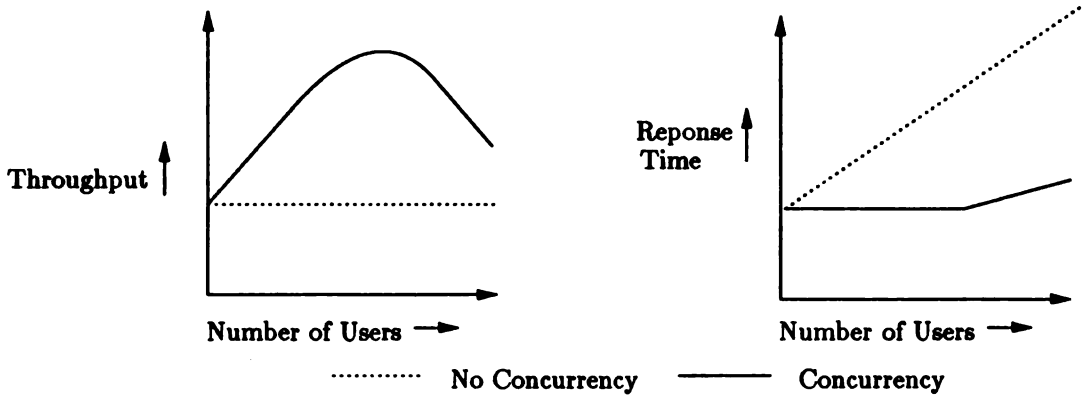


Figure 2: Concurrent processing improves performance

edge bases. We analyze its correctness and discuss how the interaction between the design of this algorithm and knowledge base features.

The organization of the paper is as follows: Section 2 contains a quick overview of concurrency control algorithms for databases and difficulties that arise in applying them to knowledge bases. In Section 3, we present a new algorithm for concurrency control for knowledge bases. We conclude the paper in Section 4 with a discussion on related work, the impact of concurrency control considerations on the design of future knowledge-based systems and a summary of the paper.

2 THE CONCURRENCY CONTROL PROBLEM

In this section, we give an example knowledge base, describe the problems caused by arbitrary interleaving and the concurrency control problems that need to be solved for knowledge bases.

2.1 AN EXAMPLE KNOWLEDGE BASE

For the purposes of this paper,[4] knowledge bases are assumed to support an object-oriented representational framework with an assertional sub-language used for both deductive rules and constraints. Also, possibly, they might support facilities for representing special kinds of knowledge (for example, temporal knowledge, incomplete knowledge, *etc.*). Where we need to talk about a specific knowledge representation notation, we will be using the language Telos (Mylopoulos et al. 1990).

As an example, consider a knowledge base that has a class **Employee** which is a specialization of class **Person**. The class **Employee** has attributes **Name**, **Manager** and **Salary**. Each **Salary** attribute indicates the salary of an employee during a certain time interval (for example, **before 1989**). **Emp01** is an instance of **Employee** and is an identifier for the employee **Adam** whose **Manager** is **John**. The salary of **Adam** takes two different values in the history — **20000** during time interval **I1** and **30000** during time interval **I2**. These intervals in the history, **I1** and **I2**, are left unspecified. There is an integrity constraint that requires that the value of the salary must always increase and, therefore, we can infer that **I1** must precede **I2**. In addition, it is assumed that the the knowledge base contains the following deductive rules:

DR1: $\forall p/\text{Employee}, \forall s/\text{Salary}, \forall t1/\text{TimeInterval}$
$\text{Salary}(p, s)(\text{at } t1) \wedge \text{GreaterThan}(s, 25000) \Rightarrow$
$\text{WellPaid}(p)(\text{at } t1)$

DR2: $\forall p/\text{Employee}, \forall t1, t2/\text{TimeInterval}$
$\text{WellPaid(p)}(\text{at } t2) \wedge \text{StartsBefore}(t2, t1) \Rightarrow$
$\text{WellPaid}(p)(\text{at } t1)$

[4] For an extensive survey, see (Brachman and Levesque 1985).

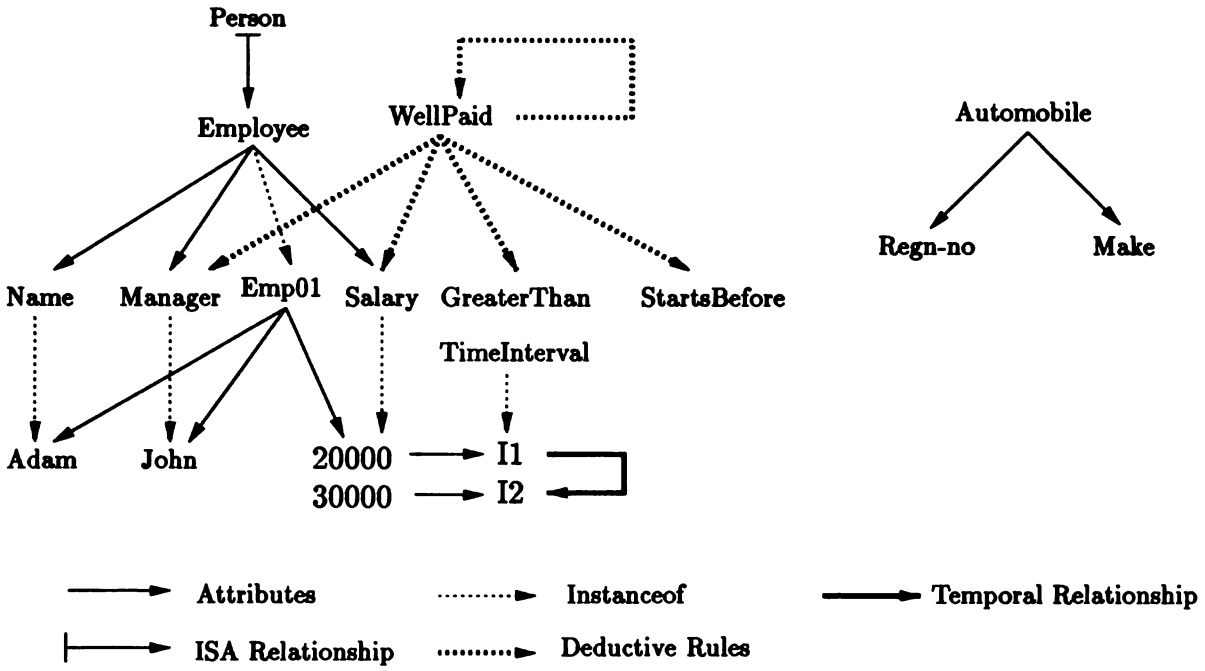


Figure 3: The knowledge base KB_1 and its structuring dimensions

The first rule states that if an employee's salary is over 25000 then the employee is well-paid. The second rule asserts that if an employee is well-paid during a time interval $t2$, she remains well-paid during any time interval that starts after $t2$. Finally, the knowledge base contains a class called **Automobile** which has attributes **Regn-no** and **Make**. For rest of the paper, we will refer to this knowledge base as KB_1.

Figure 3 shows a semantic net representation of KB_1. This knowledge base is a directed graph with five different types of edges, each corresponding to a different structuring dimension. The observation that most knowledge base features can be visualized as graphs has been the driving force in our research. As we will describe in later sections, we have taken graphs as an abstract representation for knowledge bases. We have selected our solution techniques in such a way that they are most suitable for graph structures. Such an approach helps us in understanding the interaction between knowledge base characteristics, which are abstracted in graph-theoretic terms, and the concurrency control requirements.

2.2 PERILS OF ARBITRARY INTERLEAVING

Suppose two users want to simultaneously access KB_1. The first user, called T_1, wants to change the definition of **DR1** to **DR3**, asserting that if an employee is a manager then she is well-paid:

DR3: $\forall p, q/\texttt{Employee}, \forall t1/\texttt{TimeInterval}$
$\texttt{Manager}(p, q)(\texttt{at } t1) \Rightarrow \texttt{WellPaid}(q)(\texttt{at } t1)$

The second user, called T_2, wants to find all **WellPaid** employees during the time interval **I2**.

T_1 and T_2 will perform these operations using **UNTELL**, **TELL** and **ASK** commands (Mylopoulos et al. 1990). Consider the following interleaving of their operations:

1. T_1 executes an **UNTELL** removing **DR1** from the knowledge base.

2. T_2 executes an **ASK** to find all **WellPaid** employees. Currently, the knowledge base contains only **DR2**. Therefore, T_2 gets an answer that there are no **WellPaid** employees.

3. T_1 completes its job by a **TELL** command and adds **DR3**.

If we execute the operations of T_1 and T_2 without any interleaving and in the order T_1 after T_2, we get an answer **Adam**. If the order is T_2 after T_1, we will get an answer **John**. The answer returned in the above execution does not correspond to any state of the knowledge base and, therefore, such an interleaving is *incorrect*. The notion of *correctness* of concurrent executions has been formalized through the concept of *serializability* (Eswaran et al. 1976; Bernstein et al. 1978; Papadimitriou 1979). Let *transaction* refer to the execution of a user program on a knowledge base. Two executions are *equivalent* if they leave the knowledge base in the same state, and if each operation returns the

same value in both executions. An interleaved execution of transactions is *serializable* if it is equivalent to some serial execution of the same collection of transactions. This example shows that not all concurrent interleaved executions of transactions are correct (serializable). The mechanism that controls the order in which the operations of concurrent transactions are processed, so that the overall execution is serializable, is called *concurrency control algorithm* or *policy*. The next sub-section describes some concurrency control algorithms from databases.

2.3 CONCURRENCY CONTROL ALGORITHMS FROM DATABASES

There is a vast body of literature on concurrency control algorithms (Papadimitriou 1986; Bernstein, Hadzilacos and Goodman 1987). There are three broad classes of such algorithms: locking, timestamps and serialization graphs. For each of these classes, there are variations based on multiple versions and optimistic methods. Locking-based algorithms have been most successful in practice and their performance is better understood. They also have special solutions for graph structures — the abstraction of knowledge base that appears to be the most appealing. Therefore, we have adopted the locking class of methods for knowledge bases. In the rest of the paper, we will focus on locking algorithms. The discussion on the other methods can be found in (Chaudhri, Hadzilacos and Mylopoulos 1992).

We will describe here two well-known locking algorithms. The first, known as two phase locking, does not make any assumption about the structure of the underlying data. The second, known as the DAG policy, assumes that the underlying data is structured as a directed acyclic graph.

Two-phase locking (2PL) (Eswaran et al. 1976) in simplified terms works as follows:

TP1. Associated with each data item is a distinct "lock". A transaction must acquire a lock on a data item before accessing it.
TP2. While a transaction holds a lock on a data item, no other transaction may access that data item.
TP3. A transaction cannot acquire any additional lock once it has released some lock (hence the name two-phase locking).

It can be shown that two-phase locking ensures serializability. In the example of the previous section, T_1 will lock `WellPaid` before changing its definition so that T_2 will not be able to read the partially updated value. This prevents the incorrect execution. If a transaction must acquire a lock (because of the rule TP1), but cannot do so (because of rule the TP2), it must wait until the transaction that owns that lock releases it. It is easy to construct scenarios in which locks are acquired in such a manner that a *deadlock* arises (Yannakakis 1982b): a cyclical sequence of transactions each waiting for the next to release a lock it must acquire. Such deadlocks may be resolved by choosing one of the transactions, aborting it (*i.e.*, undoing any effects it had on the knowledge base state), releasing its locks and restarting it at a later time.

The DAG policy may be specified by the following rules (Silberschatz and Kedem 1980):

D1. A transaction may begin execution by locking any item.
D2. Subsequently, it can lock an item if it has locked all the predecessors (i.e., parents) of that item in the past and is currently holding a lock on at least one of them.
D3. It may lock an item only once.

Unlike 2PL, the DAG policy is deadlock-free, that is, if transactions follow the DAG policy then a deadlock never arises. Furthermore, the DAG policy allows a transaction to release certain locks before it has acquired all locks it will ever need. The freedom of transactions to release locks earlier, often results in a greater degree of concurrency than would be possible under 2PL.

The next sub-section explores the applicability of these two locking policies to knowledge bases.

2.4 INADEQUACIES OF EXISTING METHODS AND FOCUS OF PRESENT WORK

Both 2PL and the DAG policy are possible candidates for a concurrency control algorithm for knowledge bases. However, as we describe below, both of them are inadequate for direct application to knowledge bases.

2PL requires that a transaction must hold all its locks until it has finished acquiring all of them. This has serious performance implications for the type of transactions likely to be applied to knowledge bases. For example, in the knowledge base KB_1 (Figure 3), while proving a goal through backward chaining (Genesereth and Nilsson 1987), a transaction is likely to access all the items that are below that goal in the inference graph, potentially a set including all deductive rules. Other examples of such potentially global knowledge base operations include truth maintenance systems (de Kleer 1986), temporal reasoning (Allen 1983) and recursive queries (Naqvi and Tsur 1989). In such situations, if we use 2PL, transactions will end up locking large portions of the knowledge base for long periods of time, thus significantly reducing concurrency.

The DAG policy, which holds only a small number of locks at any given time, could be a possible answer to this problem. We can use the DAG policy, if we view the underlying structure of the knowledge base

T_1= ⟨(D (WellPaid,Greaterthan)) (D (WellPaid,Salary)) (I (WellPaid,Manager))⟩

Figure 4: An example transaction

```
T1: (D (WellPaid,Greaterthan)) (D (WellPaid,Salary))        (I (WellPaid,Manager))
T2:                                                  (A WellPaid) ...
```

Figure 5: An example schedule

(for example, through the presence of generalization hierarchies or deductive rules structure) as a graph. However, the DAG policy assumes that there are no cycles in the underlying structure and the structure does not undergo any change (Yannakakis 1982a). Unfortunately, the structure of a knowledge base will contain cycles (*e.g.*, in the inference graph generated for a collection of recursive rules) and will undergo change (*e.g.*, when rule definitions are changed or rules are added or deleted). This means that the DAG policy cannot be directly applied to knowledge bases.

Thus, 2PL is not likely to give good performance for knowledge bases whereas the DAG policy does not offer enough functionality. Motivated by this, in the first phase of our research, we have extended the DAG policy to the Dynamic Directed Graph (DDG) policy that can handle cycles and updates in a knowledge base. In the next phase of our research, we plan to undertake a performance analysis of this policy which will give us guidelines for tuning the knowledge base structure for a multi-user environment and the comparative evaluation of the DDG and 2PL policies. The present paper describes the Dynamic Directed Graph (DDG) policy and its properties.

3 DYNAMIC DIRECTED GRAPH POLICY — AN ALGORITHM FOR KNOWLEDGE BASES

We begin this section by describing our assumed framework. Then, we describe the Dynamic Directed Graph (DDG) policy and its formal properties. We conclude this section by giving a locking policy that allows more concurrency than the DDG policy — but only under special circumstances.

3.1 AN ABSTRACT MODEL OF KNOWLEDGE BASES

We assume that a knowledge base is a directed graph $G = (V, E)$ where V is a set of nodes v_i (for example `WellPaid`), and E is a set of edges which are ordered pairs (v_i, v_j), of nodes (for example, `(WellPaid,Salary)`). We will use the generic term *entity* to denote both nodes and edges. A more precise representation of the knowledge base KB_1, shown in Figure 3, would have been a directed graph whose edges are of different colors, corresponding to the different types of relationships between the nodes. In the initial phase of our investigation, we do not distinguish amongst different types of edges.

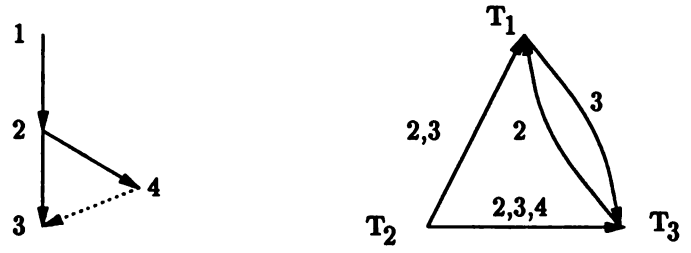
A user interacts with the knowledge base by means of transactions. Each transaction is a sequence of `TELL`, `UNTELL`, `RETELL` and `ASK` operations (Mylopoulos et al. 1990). These operations are implemented by means of more primitive operations. For example, in terms of a graph representation, the transaction T_1 of Section 2.2 consists of several primitive operations: delete the edges `(WellPaid,Salary)`, `(WellPaid,Greaterthan)` and insert the edge `(WellPaid,Manager)`.

In our model, we will represent only these primitive operations and we will consider them to be *atomic*. Formally, an operation is a pair (a e), where a is an action (one of `INSERT`, `ACCESS`, `DELETE`, abbreviated by `I`, `A` and `D` respectively) and e is an entity, which is a node or an edge. For example, the transaction T_1 of Section 2.2, that changes the definition of the rule `DR1` to `DR3`, could be specified as in Figure 4.

In addition to the operations introduced above, we also define lock and unlock operations for an entity e, denoted (`L` e) and (`U` e) respectively. (`L` e) denotes the acquisition of that lock and (`U` e) the release of that lock. A *locked transaction* is a sequence of `ACCESS`, `INSERT`, `DELETE`, `LOCK` and `UNLOCK` operations. All the locked transactions are *well-formed* in the sense that a transaction cannot perform any operation on an entity unless it holds a lock in it. It is possible to generalize our results to the case of non-well-formed policies (Yannakakis 1982a) but we make this assumption to keep the model simple and intuitive.

We say that a transaction T (or an operation (a e)) is *defined* in a knowledge base state D if it does not insert (delete, access) an entity that already exists (does not exist) in the knowledge base.

A transaction system is a finite collection τ of transactions. A *schedule* S of a transaction system τ, at some instant t, is an ordering of the steps of some transactions of τ that preserves the order of actions of each transaction. The interleaving discussed in Section 2.2, and as shown again in Figure 5, is an example of a schedule. We assume that all the schedules are *legal* in the sense that while a transaction holds a lock on some



(a) The knowledge base (b) The serializability graph $D(S_{DAG})$

T_1: (L 1) (A 1) (L 2,3) (A 2,3) (U 1,2,3)
T_2: (L 2,3,4) (I (3,4)) (U 2,3,4)
T_3: (L 2,4) (A 2,4) (U 2) (L 3) (A 3) (U 3,4)

(c) The schedule S_{DAG}

Figure 6: An example of non-serializable schedule produced by the DAG policy

entity, no other transaction may acquire a lock on that entity. A schedule is *proper* if all the steps of all the transactions are defined in the knowledge base state they are executed. A *partial schedule* S of a transaction system τ is a legal and a proper schedule of any prefixes of the transactions of τ.

In our model, the *correctness* of a schedule is equivalent to *serializability*. Serializability of a schedule can be easily decided as follows (Papadimitriou 1986). Let $\overline{e}$ denote the set of nodes in an entity e (recall that an entity is a node or an edge). Construct a directed graph $D(S)$ by associating a node v_i with each transaction T_i and including an arc (v_i, v_j) if in schedule S, T_i acts on an entity e_x before T_j does on entity e_y and $\overline{e}_x \cap \overline{e}_y \neq \emptyset$. For the sake of clarity, the arcs of $D(S)$ are labelled with the entities in $\overline{e}_x \cap \overline{e}_y$. S is serializable if and only if the digraph D is acyclic.

3.2 A NAIVE APPLICATION OF THE DAG POLICY TO DYNAMIC GRAPHS

In this section, we illustrate by means of an example that the DAG policy might produce incorrect results in face of updates to the underlying DAG structure. Figure 6(a) shows a knowledge base which is manipulated by three transactions T_1, T_2 and T_3 running by the locking rules D1-D3 of the DAG policy. The schedule S_{DAG} produced by these transactions is shown in Figure 6(c) ((L 2,3,4) is a compact representation of three operations: (L 2), (L 3) and (L 4). The other operations in this schedule should be interpreted in a similar way.). When T_1 begins execution, the edge $(4, 3)$ is non-existent. The DAG policy pre-computes the locked transaction, requiring T_1 to be holding a lock on node 2 at the time it acquires a lock on node 3. In the meantime, T_2 inserts the edge $(4, 3)$ and T_3 completes a part of its execution. If T_1 continues using the lock steps that it has pre-computed, we will get the schedule S_{DAG} as shown in Figure 6(c). The serializability graph of this schedule is shown in Figure 6(b) which contains a cycle and thus S_{DAG} is not serializable. With this motivation, let us give the description of the DDG policy which in addition to handling cycles, overcomes this problem of updates.

3.3 DESCRIPTION OF THE DYNAMIC DIRECTED GRAPH (DDG) POLICY

A *locking policy* is a collection of rules which specifies how the transactions should acquire locks. Formally, a locking policy P is a relation such that $P(T, \overline{T})$, only if transaction T is a subsequence of well-formed locked transaction $\overline{T}$.

The locking rules of the DDG policy assume that the underlying graph is always connected and has a single root[5]. In the first sub-section, we will show how any arbitrary graph is converted and maintained in this restricted form. In the second sub-section, we will specify the locking rules of the DDG policy.

3.3.1 Restricting the Knowledge Base to a Rooted and a Connected Graph

Restricting the knowledge base to a rooted and a connected graph is a two step process. First, the knowledge base has to be in this form to start with, and second, this shape has to be maintained as the knowledge base undergoes updates. The first step is implemented by pre-processing rules and the second step by structure maintenance rules. We will specify these rules and then give justifications for keeping the graph rooted and connected.

[5]Root is a node with no incoming edges.

Pre-processing Rules

The pre-processing rules are applied when the knowledge base is initially started. These rules take directed graph G of the knowledge base as input and generate a graph $\overline{G}$ as output. They compute some information which is later used by locking rules and ensure that the graph has a single root and is connected. In specifying these rules and in the rest of the paper, we will deal with a cycle by considering the strongly connected component (SCC) which contains all the nodes on that cycle.

P1. Partition G into $G_i(V_i, E_i)$ $1 \leq i \leq k$, where the underlying undirected graph of each G_i is a connected component of the underlying undirected graph of G. For each component, identify the non-trivial strongly connected components[6] G_{ij} $1 \leq j \leq b_i$. For each connected component i, identify the sources s_{ij}, $1 \leq j \leq l_i$.

P2. For each $G_i(V_i, E_i)$, add a *control node* c_i. Add edges (c_i, s_{ij}) $1 \leq i \leq k$, $1 \leq j \leq l_i$. Add another control node C and the edges (C, c_i) $1 \leq i \leq k$.

P3. Call the resulting graph $\overline{G}(\overline{V}, \overline{E})$. Thus, $\overline{V} = V \cup \{c_i | 1 \leq i \leq k\} \cup \{C\}$ and $\overline{E} = E \cup \{(c_i, s_{ij}) | 1 \leq j \leq l_i, 1 \leq i \leq k\} \cup \{(C, c_i) | 1 \leq k\}$.

For example, if we apply the above process to KB_1, we will need to add nodes c_1 and c_2, one for each connected component, and a control node C for the whole graph. $\overline{G}$ will contain the original graph G, corresponding to the knowledge base KB_1 of Figure 3 and the following edges: $(c_1, \texttt{Person})$, $(c_1, \texttt{WellPaid})$, $(c_2, \texttt{Automobile})$, (C, c_1) and (C, c_2). The resulting graph $\overline{G}$ is rooted and connected.

Structure Maintenance Rules

Insert and delete operations applied to $\overline{G}$ may cause it to become dis-connected or to acquire new sources. Furthermore, the information about the connected components of the graph needs to be updated. All this is implemented by the following rules.

M1. When a new source, s_i, is created in the connected component, G_j, add the edge (c_j, s_i)

M2. When an existing source, s_i, is removed from the component G_j, remove the edge (c_j, s_i). If the removal of s_i results in the creation of new sources, then do as in M1.

M3. Two connected components G_i and G_j are merged. This will happen if an edge (v_i, v_j) is inserted with $v_i \in G_i$ and $v_j \in G_j$. Let $s_{i1}, s_{i2}, \ldots s_{il_i}$ and $s_{j1}, s_{j2}, \ldots s_{jl_j}$ be the sources of G_i and G_j respectively. Remove c_j (and therefore the edges, (C, c_j) and (c_j, s_{jm}) $1 \leq m \leq l_j$). Recompute the sources of the merged component as: $\{s_{i,1}, s_{i,2}, \ldots s_{il_i}\} \cup \{s_{j1}, s_{j2}, \ldots s_{jl_j}\} - \{v_j\}$. Add edges (c_i, s_{ij}) for the new set of sources.

M4. A connected component $G_i(V_i, E_i)$ is split into $G_j(V_j, E_j)$ and $G_k(V_k, E_k)$. Compute new sources and add or delete appropriate edges.

M5. As there are updates in the graph, keep updating the information on connected components[7].

For example, if we want to store in KB_1, the information about the automobiles owned by an employee, we can define `Vehicle` as an additional attribute of `Employee` and make it an instance of `Automobile` (achieved by inserting a node `Vehicle` and adding the edges `(Employee,Vehicle)` and `(Automobile,Vehicle)`). This will merge the two components of the graph G. After applying M3, the new $\overline{G}$ will no longer have the control node c_2 and instead, will have the edge $(c_1$,`Automobile`). The resulting graph is still rooted and connected.

Now, let us give some justification for this process. If the graph corresponding to a knowledge base is not connected, a transaction can span more than one component. To guarantee the correctness of all the schedules in such a situation, we will have to ensure that the transactions that access some components in common follow the same serialization order in these components. This could be achieved by maintaining a graph external to the knowledge base, in which there is a node v_i corresponding to each transaction T_i, and an edge (v_i, v_j) if T_i precedes T_j in some component. Assuming the executions are serializable within each connected component, the schedules produced in such a situation will be correct if this external graph is acyclic. This external graph is not necessary if there is only one connected component in the knowledge base. Thus, if we keep the underlying graph of the knowledge base connected at all times, we save the cost of maintaining and checking cycles in this external graph. On the other hand, we incur some cost in pre-processing and structure maintenance which is comparable to the cost of maintaining the external graph. Overall, our proposed design results in a net saving, and therefore, is computationally more efficient.

In the above construction, we assume that there will be only two levels of control nodes — one level of control node for each connected component and one control node for the whole graph. The number of levels of these nodes can be varied to improve the performance of the knowledge base. The exact improvement will depend on the environment in which the knowledge

[6] A non-trivial strongly connected component is a strongly connected component that has more than one node. From now on, whenever we mention a strongly connected component, we will assume that it is non-trivial.

[7] One can use incremental graph algorithms. Such algorithms can dynamically maintain certain kinds of information about a graph in the face of updates to the graph without recomputing the information from scratch (Italiano 1986; Italiano 1988).

base will be used. We plan to explore this issue in our future research.

From now on, we will assume that the knowledge base is always connected and has a single root.

3.3.2 Acquiring Locks

The rules presented in this section are the core of the DDG policy as they specify how the transaction should acquire locks.

There are two key differences between the locking rules of the DAG policy and the DDG policy. First, the locking rules of the DDG policy are applied to the current state of the graph, whereas the locking rules of the DAG policy are applied to the state of the graph when the transaction begins execution. In case of the DAG policy, there is no need to distinguish between the initial and the current state of the graph, because the graph never changes. As we saw in the schedule S_{DAG} of the previous sub-section, this results in an undesirable behaviour in face of the updates. Second, the locking rules of the DDG policy provide a solution for cycles. It is easy to see that using the locking rules of the DAG policy, it is not possible to lock any node on a cycle. The DDG policy solves both of these problems.

Let us first give some definitions and then specify the locking rules.

Let $R(T)$ be the set of nodes to be accessed by a transaction T. Let us define a *dominator* of a set of nodes U in a rooted and connected graph to be a node d, such that all paths from the root node to each node $v \in U$ pass through d. The root node dominates all the nodes in the graph, including itself. *Entry point* of a strongly connected component (SCC), G_{ij}, is a node v of G_{ij}, such that, there is an edge (w, v) of $\overline{G}$ so that w is not in G_{ij}.

Locking Rules

L1. The first node to be locked by T is D, a dominator of $R(T)$ with respect to $\overline{G}$.

L2. Before T performs any operation (`INSERT/DELETE` or `ACCESS`), on a node v (or an edge (u, v)), T has to lock v (both u and v).

L3. A node v can be locked if and only if all its predecessors in the present state of $\overline{G}$, that do not lie on the same non-trivial strongly connected component as v, have been locked by the transaction in the past, and the transaction is presently holding a lock on at least one of them. All the nodes on a strongly connected component are locked together in one step, provided all the entry points of that SCC have been locked. A node that is being inserted can be locked at any time.

L4. Each node can be locked at most once.

For example, the transaction T_1 of Figure 4 would start by locking `WellPaid`. Then it would lock `Salary` and then the edge (`WellPaid`,`Salary`). After deleting this edge, T_1 could release the lock on `Salary` and proceed to lock `Greaterthan`. Thus, the transaction is able to acquire locks even after releasing some of the locks — a clear improvement over two-phase locking.

In the example of Figure 6, T_1 will not be allowed to lock node 3 in the new state of the knowledge base unless it has locked node 4 as well. This will prevent the incorrect schedule S_{DAG}. In the next sub-section we will prove that this is true for any schedule produced by the DDG policy.

This completes the description of the DDG policy.

3.4 PROPERTIES OF THE DYNAMIC DIRECTED GRAPH (DDG) POLICY

We will begin this section by showing that that the DDG policy always produces correct schedules. Then, we will prove that the DDG policy is deadlock-free and well-structured.

3.4.1 Correctness

The correctness of a locking policy can be formalized by the notion of *safety*.

A locked transaction system is *safe* if any legal schedule S of it is correct. A locking policy P is safe if for any transaction system $\tau = \{T_1, \ldots, T_m\}$ and $\overline{\tau} = \{\overline{T}_1, \ldots, \overline{T}_m\}$ where $P(T_i, \overline{T}_i)$ for all $1 \leq i \leq m$, the locked transaction system $\overline{\tau}$ is safe.

We will first state a theorem that characterizes the unsafe transaction systems. Intuitively, this theorem says that if a locked transaction system is not safe then it is *always* possible to construct a *canonical* non-serializable schedule, which is legal and proper, and in which all transactions except one are executed serially. This result is the generalization of a similar result for transaction systems that do not contain insert/delete operations (called static systems) (Yannakakis 1982a) to systems that contain insert/delete operations (called, dynamic systems). For the static systems, the structure of the serialization graph corresponding to the canonical schedule is linear whereas, for the dynamic systems the corresponding graph is not necessarily linear — it can be a general graph. This characterization of canonical schedules is a very useful tool in proving the correctness of the policy.

The *interaction graph* $G(\tau)$ of a transaction system τ is an undirected graph with one node T_i corresponding to each transaction of τ and an edge between any two nodes whose corresponding transactions have an entity in common. We can now state the theorem on canonical schedules.

Theorem 1 *A transaction system τ is not safe if and only if there are transactions $T_1, \ldots, T_k$ in τ ($k > 1$) and entities $A_1, \ldots, A_k$, not necessarily distinct, such that*

(1) A subsequence of $T_1, \ldots, T_k$ forms a chordless cycle in the interaction graph $G(\tau)$.

(2) In T_1 the entity A_k is locked after A_1 is unlocked.

(3a) The following partial schedule is legal and proper. Let T_1' be the prefix of T_1 up to the (L A_k) step, and T_i', for $i \neq 1$, the prefix of T_i up to and including the (U A_i) step. S' is the serial execution of $T_1', \ldots, T_k'$ in this order.

(3b) Furthermore, S' can be extended to a complete schedule, that is, can avoid deadlock.

The details of the proof of this theorem can be found in (Chaudhri, Hadzilacos and Mylopoulos 1992). □

Theorem 2 *Dynamic Directed Graph policy is a safe policy.*

Proof Outline: Suppose it is not. Then, choose transactions $T_1, \ldots, T_k$, entities $A_1, \ldots, A_k$ and the corresponding schedule S' as in Theorem 1.

The serialization graph $D(S')$ of the schedule up to (but not including) the (L A_k) step of T_1 is an acyclic graph over $T_1, \ldots, T_k$. Let d be the length of the shortest path from T_i to T_k in the serialization graph $D(S')$. Let B_i be the first entity locked by the transaction T_i and G_i be the state of the graph when T_i begins execution. By induction on d, it can be shown that for $1 \leq i \leq k$, B_i is a descendant of A_k in G_i. Let G_{k+1} be the state of the graph after T_k finishes its execution. Then, B_1 is a descendant of A_k in G_1, which is a contradiction to the fact that T_1 holds a lock on a parent of A_k in G_{k+1}. Hence the assumed non-serializable schedule as claimed does not exist and the DDG policy is safe. □

3.4.2 Deadlock-freedom

As described in Section 2.3, a deadlock is a situation when a cyclical sequence of transactions are each waiting for the next to release a lock it must acquire (Yannakakis 1982b). Such deadlocks may be resolved by choosing one of the transactions, aborting it (*i.e.*, undoing any effects it had on the knowledge base state), releasing its locks and restarting it at a later time. Thus, if a locking policy is deadlock-free, it would mean that it will never have to abort any transaction unnecessarily.

Let us consider a typical case of deadlock for a set τ of transactions. Deadlock arises in a partial schedule S of τ when every transaction wants to lock an entity in the next step that is already locked by some other transaction. This means that there is a set of transactions $\{T_1, \ldots, T_k\}$ such that the next step of T_i is (L x_i) where x_i is currently locked by T_{i+1} (where we take $k + 1 = 1$).

Thus, in the partial schedule S, transaction T_i accesses x_{i-1} before T_{i-1}; if S could possibly finish in any way then the resulting schedule would not be correct. In other words, deadlocks prevent some wrong schedules from finishing. Let us show that the DDG policy is deadlock-free using this fact.

Theorem 3 *The Dynamic Directed Graph (DDG) policy is deadlock-free.*

Proof Outline: In the deadlock state, none of the transactions can release the locks on entities that are causing the deadlock. This means that these entities must be parents of the entities that need to be locked. But this would imply that these entities lie on a cycle. The DDG policy locks a cycle in one step which could not be locked by different transactions simultaneously. Hence, the DDG policy must be deadlock-free. □

3.4.3 Well-structured-ness

A locking policy P is *well-structured* if it allows a transaction to access an arbitrary set of entities in the knowledge base. Formally, P is well-structured if for any set of entities D in the knowledge base and any transaction T, such that T is defined in D[8], there is a locked transaction $\overline{T}$ such that $P(T, \overline{T})$ and $\overline{T}$ is also defined in D.

Theorem 4 *The DDG policy is well-structured.*

Proof Outline: It will be possible for T to lock an arbitrary set of entities in the underlying graph if we can identify one entity by starting from which the locking rule L3 will be satisfied for all the entities in the transaction. Since the rules P1-P3 and M1-M5 ensure that the graph is always connected and has a single source, the node C of $\overline{G}$ satisfies this property for all the nodes in the graph. Hence, the DDG policy is well-structured. □

3.5 A MORE LIBERAL VARIANT OF THE DYNAMIC DIRECTED GRAPH (DDG) POLICY

The locking rule L3 of the DDG policy requires that once all entry points have been locked, nodes of a strongly connected component (SCC), and therefore, all the nodes on a cycle, should be locked together in one step. This will not permit any concurrency within

[8] Recall that a transaction T is defined in a knowledge base state D if it does not insert (delete, access) an entity that already exists (does not exist) in the knowledge base.

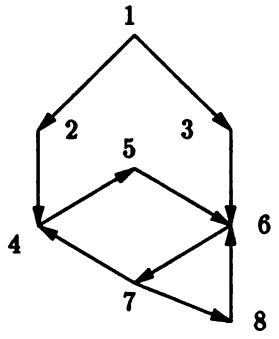


Figure 7: Permitting concurrency within cycles

the cycles. A natural question is whether this is the best one can do or is there a locking policy that can allow concurrency within cycles. We present a rule L3′ which permits concurrency within an SCC.

L3′ If a node does not lie on an SCC or it is not an entry point of an SCC, it can be locked if and only if all its predecessors in the present state of $\overline{G}$ have been locked by the transaction in the past and the transaction is presently holding a lock on at least one of them.

If a node is an entry point of an SCC, it can be locked if and only if all its predecessors in the present state of $\overline{G}$, that do not lie on the same SCC, have been locked by the transaction in the past and the transaction is presently holding a lock on at least one of them.

Replacing L3 by L3′ in the DDG policy gives a locking policy DDG′, which allows concurrency within cycles. For example, consider a graph G, shown in Figure 7. This graph has one non-trivial strongly connected component namely 4,5,6,7,8. The DDG′ policy will permit transactions $T_1 = \langle(\mathbf{A}\ 2,4,5)\rangle$ and $T_2 = \langle(\mathbf{A}\ 3,6,7)\rangle$ to concurrently access this SCC. However, transactions $T_3 = \langle(\mathbf{A}\ 2,4,5,6)\rangle$ and $T_2 = \langle(\mathbf{A}\ 3,6,7,4)\rangle$ cannot concurrently access this SCC. Each of them will begin execution by locking the node 1. Suppose, T_3 locks the node 1 before T_4 locks it. T_3 will be able to lock the entry points 4 and 6 before T_4 and it will not release locks on them until it has finished processing the whole SCC. Thus, T_3, will be the only transaction executing within the SCC (Similar arguments can be made when T_4 locks node 1 before T_3.). Thus, the DDG′ policy permits concurrency within an SCC to a limited extent. The DDG′ policy satisfies the same properties as the DDG policy:

Theorem 5 *The DDG′ policy is a safe, deadlock-free and well-structured policy.*

Proof Outline: A simple generalization of the proofs for the DDG policy works for the DDG′ policy. □

Let us analyze the relative merits of the DDG and the DDG′ policies. The DDG policy is intuitive and simple as it treats strongly connected components (and cycles) as a unit of locking. In the presence of updates, the DDG′ policy will require more bookkeeping effort than the DDG policy, and therefore, the DDG policy appears more suitable for the case of updates. Furthermore, in knowledge base applications such as recursive rules, it is highly likely that a transaction will access *all* the entities on a cycle and probably access them *more than once*. Since, a transaction is allowed to lock an entity only once[9] (L4), it means that it will have to retain all the locks on an SCC until it has finished processing it. Thus, if an SCC has only one entry point, then there is no difference in the concurrency permitted by the two policies. Furthermore, when an SCC has more than one entry point, the concurrency within an SCC is possible only if some nodes can be accessed through different entry points. Therefore, we feel that, in general, we cannot take advantage of the more concurrency allowed by the DDG′ policy. Based on these arguments, we have adopted the DDG policy as our initial design.

4 RELATED WORK, CONCLUSIONS AND SUMMARY

There has been very little work on concurrency control with a specific reference to knowledge bases. (Raschid, Sellis and Lin 1988) uses concurrency to improve the performance of rule execution in the context of the OPS5 system (Forgy 1982). Their proposal is to parallelize the inference process of one user, whereas our focus is on parallelizing the operations of different users. Similar efforts have been made in (Filman 1989; Ishida, Yokoo and Gasser 1990; Schmolze and Goel 1990).

(Garza and Kim 1988) studies concurrency control problem for object-oriented databases and proposes a locking method that has provisions for variable units of locking. (Elkan 1990) looks at deductive databases and proposes efficient algorithms for checking conflicts between transactions. Neither of these papers addresses the problem that the large portions need to be locked in knowledge bases, and that the internal structure of a knowledge base may be used to do more efficient concurrency control. There is a flurry of work on flexible transaction models (Berghouti and Kaiser 1991) focusing on the problems that arise in domains such as software engineering and CAD.

Multidatabase concurrency control looks at the situations when a transaction can span more than one system (Breitbart, Garcia-Molina and Silberschatz 1992). This research will be useful for knowledge bases because we can easily visualize situations when a part of a

[9]This condition is necessary to guarantee the safety of a locking policy (Yannakakis 1982a).

transaction is executed in a knowledge base (inference) and a part in a database (data retrieval) (Brodie 1989). We do not address the problem of multidatabase transactions in our research.

According to our knowledge, concurrency control has not received the attention of AI community.

The design of locking policy for cycles is an example of how the knowledge base features might interact with techniques such as concurrency control. The discussion in Section 3.5 illustrates that, in general, our locking policy will not allow transactions to concurrently access entities along a cycle. This means that for performance reasons, we should try to design a knowledge base with few and small cycles. This suggests the existence of an *expressiveness vs performance* tradeoff meaning that good performance considerations dictate ruling out certain knowledge base designs (for example, a knowledge base with too many large cycles).

The expressiveness vs performance tradeoff is analogous to the *expressiveness vs tractability* tradeoff (Levesque and Brachman 1985) where complexity considerations dictate ruling out certain knowledge base designs. Performance, measured in terms of throughput and response time (Lazowska et al. 1984), is a measure of desirability, similar and related to complexity — but may be of more pragmatic nature. Furthermore, the designs ruled out by the complexity considerations are not likely to be the same as those ruled out for performance reasons. For example, negations cause intractability in the reasoning (Levesque and Brachman 1985; Nebel 1988), but it will be possible to view the knowledge base containing negations as a graph and use the algorithm presented in this paper. On the other hand, the reasoning with cycles is tractable but the presence of cycles has a potential of causing performance bottlenecks. We suggest that, in addition to the complexity considerations, performance requirements should be a design criteria for knowledge-based systems of future.

There seem to be many interesting extensions to this work which we believe will affect the way we think about the design of knowledge-based systems. For example, we are beginning work on the development of a performance model that will tell us the desired features of a graph for the best performance. This will give us recommendations on structuring of a knowledge base for optimal performance in a multi-user environment. We are planning to generalize this algorithm to handle graphs with edges of different "colors" and to have more general locking modes. We are also considering to implement this algorithm as part of a knowledge base so as to verify its applicability in a knowledge base environment. Further down the road, we expect that issues of fault tolerance such as recovery (Bernstein, Hadzilacos and Goodman 1987) will play an important role in the extension of our research results.

To summarize, this paper has argued that concurrency control is a necessity for large multi-user knowledge bases. The paper then focused on a locking-based approach that seems most viable from a practical viewpoint. A locking algorithm, called the Dynamic Directed Graph (DDG) policy, was presented that exploits the structure of a knowledge base and selectively locks only a small number of entities at any one time. The DDG policy can handle situations where the knowledge base contains cycles and undergoes changes over time. Our results include a proof of correctness of the proposed algorithm, an analysis of its properties and a discussion of how concurrency control requirements might affect the knowledge-based systems of the future.

In conclusion, we would like to claim that the paradigms for the design of knowledge-based systems should include *expressiveness vs performance* tradeoffs, where performance is measured in terms of throughput and response time. We believe that algorithms such as the ones presented in this paper constitute a modest contribution towards this direction and can provide an important enabling technology to the goal of knowledge sharing, as articulated in (Neches et al. 1991).

Acknowledgments

This research was supported by the University of Toronto, the Information Technology Research Centre of Ontario, the Natural Science and Engineering Research Council of Canada and the Institute of Robotics and Intelligent Systems.

References

Allen, J. F. 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843.

Berghouti, N. S. and Kaiser, G. E. 1991. Concurrency Control in Advanced Database Applications. *Computing Surveys*, 23(3):269–318.

Bernstein, P. A., Hadzilacos, V., and Goodman, N. 1987. *Concurrency Control and Recovery in Database Systems.* Addison-Welssley Publishing Company.

Bernstein, P. A., Rothnie, J. B. J., Goodman, N., and Papadimitriou, C. H. 1978. The Concurrency Control Mechanism of SDD-1: A System for Distributed Databases (The Fully Redundant Case). *IEEE Transactions on Software Engineering*, 4(3):154–168.

Brachman, R. J. and Levesque, H. J., editors 1985. *Readings in Knowledge Representation and Reasoning.* Morgan Kaufmann Publishers, Inc., Los Altos, CA.

Breitbart, Y., Garcia-Molina, H., and Silberschatz, A. 1992. Overview of Multidatabase Transaction Management. Technical Report STAN-CS-92-1432, Stanford University.

Brodie, M. L. 1989. Future Intelligent Information Systems: AI and Database Technologies Working Together. In Mylopoulos, J. and Brodie, M. L., editors, *Artificial Intelligence and Databases*. Morgan Kaufmann Publishers Inc., CA.

Chaudhri, V. K., Hadzilacos, V., and Mylopoulos, J. 1992. Concurrency Control for Knowledge Bases. Technical Report Forthcoming, University of Toronto.

de Kleer, J. 1986. An assumption-based TMS. *Artificial Intelligence - An International Journal*, 28(2):127–162.

Elkan, C. 1990. Independence of Logic Database Queries and Updates. In *Proceedings of the 1990 Principles of Database Systems Conference*, pages 154–160.

Eswaran, K., Gray, J. N., Lorie, R. A., and Traiger, I. L. 1976. The Notions of Consistency and Predicate Locks in Database Systems. *Communications of the ACM*, 19(9):624–633.

Filman, R. E. 1989. New Generation Knowledge System Development Tools. Technical Report DARPA Contract No. F30602-85-C-0065, IntelliCorp Inc.

Forgy, C. L. 1982. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence - An International Journal*, 19(1):17–37.

Garza, J. F. and Kim, W. 1988. Transaction Management in an Object-Oriented Database Systems. In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, pages 37–45, Chicago, IL.

Genesereth, M. R. and Nilsson, N. J. 1987. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, Los Altos, CA.

Gray, J. N. 1992. Personal Communication.

Ishida, T., Yokoo, M., and Gasser, L. 1990. An Organizational Approach to Adaptive Production Systems. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 52–58, Boston, MA.

Italiano, G. F. 1986. Amortized Efficiency of a Path Retrieval Data Structure. *Theoretical Computer Science*, 48(2,3):273–281.

Italiano, G. F. 1988. Finding Paths and Deleting Edges in Directed Acyclic Graphs. *Information Processing Letters*, 28(1):5–11.

Lazowska, E. D., Zahorjan, J., Graham, G. S., and Sevcik, K. C. 1984. *Quantitative System Performance: Computer System Analysis using Queueing Network Models*. Prentice-Hall, Inc., Englewood Cliffs, NJ.

Lenat, D. B. and Guha, R. 1989. *Building Large Knowledge-based Systems: Representation and inference in the Cyc Project*. Reading, Mass.:Addison-Wesley Publishing Co.

Levesque, H. J. and Brachman, R. J. 1985. A Fundamental Tradeoff in Knowledge Representation and Reasoning. In Brachman, R. J. and Levesque, H. J., editors, *Readings in Knowledge Representation*, pages 42–70. Morgan Kaufmann Publishers, CA.

Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M. 1990. Telos: A Language for Representing Knowledge About Information Systems. *ACM Transactions on Information Systems*, 8(4):325–362.

Naqvi, S. and Tsur, S. 1989. *A Logical Language for Data and Knowledge Bases*. Computer Science Press, New York.

Nebel, B. 1988. Computational Complexity of Terminological Reasoning in BACK. *Artificial Intelligence - An International Journal*, 34(3):371–383.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., and Swartout, W. 1991. Enabling Technology for Knowledge Sharing. *AI Magazine*, 12(3):36–56.

Papadimitriou, C. 1979. Serializability of Concurrent Database Updates. *Journal of the Association for Computing Machinery*, 26(4):631–653.

Papadimitriou, C. 1986. *The Theory of Database Concurrency Control*. Computer Science Press, Rockville, MD.

Raschid, L., Sellis, T., and Lin, C.-C. 1988. Exploiting Concurrency in a DBMS Implementation for Production Systems. In *Proceedings of the International Symposium on Databases in Parallel and Distributed Systems*, pages 34–45, Austin, TX.

Schmolze, J. G. and Goel, S. 1990. A Parallel Asynchronus Distributed Production System. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 65–71, Boston, MA.

Silberschatz, A. and Kedem, Z. M. 1980. Consistency in Hierarchical Database Systems. *Journal of the Association for Computing Machinery*, 27(1):72–80.

Yannakakis, M. 1982a. A Theory of Safe Locking Policies in Database Systems. *Journal of the Association for Computing Machinery*, 29(3):718–740.

Yannakakis, M. 1982b. Freedom from Deadlock of Safe Locking Policies. *SIAM Journal of Computing*, 11(2):391–408.

XI.

Invited Talks and Panels

The DARPA Knowledge Sharing Effort: Progress Report

Ramesh S. Patil
USC Info. Sci. Inst.
Marina del Rey, California

Richard E. Fikes
Stanford University
Palo Alto, California

Peter F. Patel-Schneider
AT&T Bell Labs
Murray Hill, New Jersey

Don Mckay
Paramax Systems Corp.
Paoli, Pennsylvania

Tim Finin
Univ. of Maryland
Baltimore, Maryland

Thomas Gruber
Stanford University
Palo Alto, California

Robert Neches
USC Info. Sci. Inst.
Marina del Rey, California

Building knowledge-based systems today usually entails constructing a new knowledge base from scratch. Even if several groups of researchers are working in the same general area, such as medicine or electronic diagnosis, each team must develop its own knowledge base from scratch. The cost of this duplication of effort has been high and will become prohibitive as we build larger and larger systems. Furthermore, lack of methodology for sharing and communicating knowledge poses a significant road-block in developing large multi-center research projects such as DARPA/Rolm Laboratory Planning and Scheduling Initative [21]. To overcome these barrier and advance the state of the art, we must find ways of preserving existing knowledge bases, and sharing, reusing, and building on them.

The Knowledge-Sharing Effort, sponsored by the Defense Advanced Research Projects Agency (DARPA), The Air Force Office of Scientific Research (AFOSR), the Corporation for National Research Initiative (NRI), and the National Science Foundation (NSF), is an initiative to develop the technical infrastructure to support the sharing of knowledge among systems. [27] The goal of this effort is to develop a technology that will enable researchers to develop new systems by selecting components from library of reusable modules and assembling them together. Their effort will be focused on creating specialized knowledge and reasoners specific to the task of their system. Their new system would inter-operate with existing systems, using them to perform some of its reasoning. In this way, declarative knowledge, problem solving techniques and reasoning services could all be shared among systems. The reusable modules in the library them-selves will benefit from refinements that are only possible through extensive use. This would facilitate building larger systems cheaply and reliably. The infrastructure to support such sharing and reuse would lead to greater ubiquity of these systems, potentially transforming the knowledge industry.

The work in the Knowledge-Sharing Effort began with the identification of the impediments to knowledge sharing and corresponding needs for the development of technology to overcome these impediments. Four key areas were identified for the initial effort. They are: (1) mechanisms for translation between knowledge bases represented in different languages; (2) common versions of languages and reasoning modules within families of representational paradigm; (3) protocols for communication between separate knowledge-based modules, as well as between knowledge-based systems and databases; and, (4) libraries of "ontologies," i.e., pre-fabricated foundations for application-specific knowledge bases in a particular topic area.

A detailed discussion of the impediments, and an analysis of the issues that motivated us to focus on these four types, appears in [27]. That article also describes the working groups (comprised of researchers from the DARPA AI community and other volunteers) that were established to address these issues. The next four sections describe the progress made by each of the four working groups in addressing these issues through the development of draft specifications, implementations and experiments.

1 AN INTERLINGUA FOR KNOWLEDGE INTERCHANGE

For a knowledge-based system to incorporate encoded knowledge from a library or to interchange knowledge with another system, the knowledge must either be represented in the receiving system's representation language or be translatable in some practical way into that language. Since an important means of achieving efficiency in application systems is to use specialized representation languages that directly support the knowledge processing requirements of the application, we cannot expect a standard knowledge representation language to emerge that would be used generally in application systems. Thus, we are confronted with a *heterogeneous language problem*. We may, however, be able to deal with that problem by developing a knowledge interchange language that would be commonly used as an *interlingua* for communicating knowledge

between computer programs.

Given such an interlingua, a sending system could translate knowledge from its application-specific representation into the interlingua for communication purposes and a receiving system could translate knowledge from the interlingua into its application-specific representation before use. In addition, the interlingua could be the language in which libraries would provide reusable knowledge bases. An interlingua eases the translation problem in that without an interlingua one must write N pairs of translators in order to communicate knowledge to and from N other languages. With an interlingua, one need only write one pair of translators into and out of the interlingua.

1.1 KIF – A KNOWLEDGE INTERCHANGE FORMAT

The Interlingua Working Group, chaired by Richard Fikes and Michael Genesereth, is attacking the heterogeneous language problem by developing and testing a language for use as an interlingua called the Knowledge Interchange Format (KIF)[16]. The group began its work by observing that an interlingua needs to be a language with the following general properties:

- A formally defined declarative semantics;
- Sufficient expressive power to represent the declarative knowledge contained in typical application system knowledge bases; and
- A structure that enables semi-automatic translation into and out of typical representation languages.

The working group then merged ongoing language design efforts to produce a preliminary version of the KIF language which could be used as a straw man interlingua in knowledge interchange experiments and design discussions. Since then, the language has been continually evolved and expanded based on feedback from ongoing e-mail discussions, formal design reviews, translation of example knowledge bases, and interoperation experiments.

KIF is an extended version of first order predicate logic. The current 3.0 version of KIF has the following features:

- Simple list-based linear ASCII syntax suitable for transmission on serial media. For example, the following is a KIF sentence:

```
(forall ?x (=> (P ?x) (Q ?x)))
```

- Model-theoretic semantics with axiomatic characterization of a large vocabulary of object, function, and relation constants.
- Function and relation vocabulary for numbers, sets, and lists.
- Support for expression of knowledge about the properties of functions and relations. Functions and relations are included in the universe of discourse as sets of lists so that they can be arguments to relations (e.g, **transitive** and **one-one**) and functions (e.g., **inverse** and **range**). In addition, a **holds** relation is included that is true when its first argument denotes a relation that has as a member the list consisting of the items denoted by the remaining arguments. So, for example, one could define transitivity as follows:

```
(<=> (transitive ?r)
     (=> (holds ?r ?x ?y)
         (holds ?r ?y ?z)
         (holds ?r ?x ?z)))
```

- A sublanguage for defining objects, n-ary relations, and n-ary functions that enables augmentation of the representational vocabulary and specification of domain ontologies. Definitions can be *complete* in that they specify an equivalent expression or *partial* in that they specify an axiom that restricts the possible denotations of the constant being defined. For example, the following is a complete definition of the unary relation **bachelor**:

```
(defrelation bachelor (?x) :=
   (and (man ?x) (not (married ?x))))
```

and the following is a partial definition of a binary relation **above** which specifies that **above** is transitive and holds only for "located objects":

```
(defrelation above (?b1 ?b2)
   :=> (and (located-object ?b1)
            (located-object ?b2))
   :axiom (transitive above))
```

- Support for expression of knowledge about knowledge. KIF expressions are included as objects (i.e., lists) in the universe of discourse, and functions are available for changing level of denotation. For example, the following sentence says that Lisa has the same belief as John about the material of which things are made:

```
(=> (believes john '(material ,?x ,?y))
    (believes lisa '(material ,?x ,?y))
```

and the following sentence says that every sentence of the form (=> ϕ ϕ) is true:

```
(=> (sentence ?p) (true '(=> ,?p ,?p)))
```

- A sublanguage for stating both monotonic and nonmonotonic inference rules. For example:

```
(<<= (flies ?x)
     (bird ?x) (consis (flies ?x)))
```

A KIF reference manual describing the entire language in detail is available through anonymous FTP from

hudson.stanford.edu[17]. The working group expects the current language design to remain relatively stable and for future versions to be essentially extensions to the existing language. Extensions under active consideration include support for uncertain knowledge and contexts, and additional support for default knowledge.

KIF is intended to be a core language which is expandable by defining additional representational primitives. For example, one can define a frame language vocabulary of classes, slots, number restrictions, value restrictions, etc. (as Gruber has done in [19]) so that knowledge can be expressed in a form directly analogous to a frame language. Thus, given suitable definitions, one could define a "guest meal" as being a meal in which there is at least one guest and the food is gourmet as follows:

```
(defrelation guest-meal (?m)
   :=> (and (meal ?m)
            (at-least-fillers ?m guest 1)
            (all-fillers ?m food
                         gourmet-food)))
```

1.2 KNOWLEDGE INTERCHANGE EXPERIMENTS USING KIF

The problems involved in interchanging knowledge bases are not yet well understood, and there is open debate as to whether a generally useful interlingua can be specified. The Interlingua Working Group is attempting to inform that debate by developing KIF as a candidate interlingua and by promoting knowledge interchange experiments designed to substantially test the viability and adequacy of KIF as an interlingua. Several small scale experiments have been conducted thus far and multiple projects are underway to build and test KIF translators. These activities, though still in preliminary stages, have already been very productive in identifying issues that need to be resolved and technology that needs to be developed in order for knowledge interchange to be a practical reality. We describe three examples of such activities below.

Ramesh Patil built translators to an early version of KIF from CLASSIC [4] and from LOOM [22]. He then used those translators to produce KIF versions of simple CLASSIC and LOOM knowledge bases. As expected, such translation experiments highlighted weaknesses in KIF and motivated evolution of the language. In general, producing KIF translations of a wide range of sample knowledge bases is an effective means of evaluating the expressive adequacy of KIF and focusing its continuing development. Building the translators themselves does not appear to be problematical. The primary issue is whether KIF has sufficient expressive power to represent the declarative knowledge expressible in the source language.

Translating knowledge *out of* KIF is in general an intractable problem because any given proposition can be expressed in KIF in many equivalent but syntactically different forms and the recognition grammar for a target language will only be able to recognize some subset of those forms. The translation task, therefore, involves applying equivalence preserving rewrite rules to transform unrecognizable sentences into recognizable forms. Despite the worst-case complexity of logically complete translation, effective translation may be achievable in most situations by logically incomplete techniques combined with interactive direction from the user. To explore that hypothesis, Fikes and Van Baalen are building a translator development "shell" which will contain a grammar-based recognizer, a goal-directed rewrite rule interpreter, a library of general-purpose rewrite rules, facilities for hand translation of problematic sentences, etc. [12]. Initial versions of the basic components of that shell have been implemented and have been used to successfully translate simple KIF knowledge bases into CLASSIC.

A knowledge interchange capability is important both to enable *incorporation* of knowledge into a knowledge-based system (e.g., during system development) and to enable *interoperation* of knowledge-based systems so that they can cooperatively perform tasks and solve problems. KIF is being used as the knowledge level inter-agent communication language in multiple interoperation experiments, including those conducted by Mike Genesereth using the Designworld system [15] and those being conducted by participants in the Palo Alto Collaborative Testbed (PACT).

Designworld is an automated prototyping system for small scale electronic circuits built from standard parts (TTL chips and connectors on prototyping boards). The design for a product is entered into the system via a multi-media design workstation; the product is built by a dedicated robotic cell; and, if necessary, the product, once built, can be returned to the system for diagnosis and repair. The system consists of eighteen processes on six different machines. Each of the eighteen programs is implemented as a distinct agent that communicates with its peers via messages in a KQML-like Agent Communication Language (ACL) that uses KIF as the "content" language.

PACT is a laboratory for exploring the use of knowledge sharing technology and agent-based system integration architectures to support concurrent engineering. Participants include research groups at Stanford University, Lockheed AI Laboratory, Hewlett-Packard Laboratories, and Enterprise Integration Technologies. The initial experiments integrated four preexisting concurrent engineering systems into a common computational framework and explored engineering knowledge exchange in the context of a distributed simulation and simple incremental redesign scenario [9]. In those experiments, each of the individual systems was

used to model one or more components of an example programmable electro-mechanical device, a small robotic manipulator. The systems interact via software agents which use KQML as the "performative" language and KIF as the "content" language during knowledge interchange.

Although these experiments have not yet placed severe demands on KIF as an interlingua, KIF successfully provided what was needed, namely a clearly specified logical sentence language for interchange of assertions, queries, and simulation inputs and outputs.

2 THE KNOWLEDGE REPRESENTATION SYSTEM SPECIFICATION

Even within a single family of knowledge representation systems (e.g. KL-ONE) minor differences in syntax and semantics between systems pose significant barriers to knowledge sharing. The goal of the Knowledge Representation System Specification (KRSS) group is to develop common specifications for the representational component of families of knowledge representation systems. These specifications will help facilitate the transfer of collections of knowledge between knowledge representation systems in the same family, by reducing the representational differences among systems in the family. The intent of the group is to produce, by-and-large, descriptive specifications, although reconciliation of some syntactic differences will almost certainly be required.

Specifications produced by the group will concentrate on the representational components of the family of knowledge representation systems. Thus, they will provide a complete definition of the representation language underlying the family, but will not include a complete definition of the interface functions that are required in a useful knowledge representation system. Instead the specifications will only define a minimal interface, one that is sufficient to create knowledge bases and query them in limited ways. Also, specifications will completely ignore user-interface issues.

These specifications will definitely not be interlinguas. The representation formalism in the specifications will be specific to the family of representation systems under consideration, and will not be general-purpose representation logics. The specifications also have to be concerned with the computational properties of the formalism they define (i.e., how hard inference in the formalism is), as the aim of the group is to specify knowledge representation *systems*, and not just abstract formalisms.

The initial effort of the KRSS group is the development of a specification for knowledge representation systems based on what are now called description logics (also known as frame-based description languages, terminological logics, etc.). These systems include BACK [31], CLASSIC [6], KRIS [2], and LOOM [22]. This group of systems was chosen partly because there is a large number of systems that are based on description logics (see above), partly because there was already some interest in the community of developers of such systems for a common specification [1], partly because many of the people in the initial group gathered together at the start of the DARPA Knowledge Sharing Initiative were working with such systems, and partly because such systems have a formal basis that is readily amenable to a well-defined specification. There has also been considerable study of the formal properties of reasoning in systems based on description logics. This includes studies of how reasoning should proceed in such systems [26] and the computational complexity and decidability of reasoning in description logics [5, 25, 10, 30]. The presence of such a large body of formal work makes the specification process much easier.

Although there is a common background for all knowledge representation systems based on description logics, there is surprising variance in several dimensions in the systems. First, different systems have different input syntaxes. One goal of the initial KRSS effort is to minimize differences in this dimension. Second, different systems have different interfaces, both functional and user interfaces. Another goal of the initial KRSS effort is to minimize differences in the portion of the functional interface used to construct and directly query knowledge bases. However, the rest of the interfaces of the various systems will not be incorporated into the specification, as it is outside the goals of the KRSS group.

The main difference between the various systems is that they take different positions in the trade-offs among expressive power, completeness of inference, and resource consumption. Some systems try to be as complete as possible in a less-expressive description logic while consuming as few resources as possible, trading off expressive power for computational benefits. Some systems implement complete inference in a moderately-expressive but decidable description logic, trading off possible resource consumption for better expressive power. Some systems implement only partial inference in an expressively-powerful description logic, trading off completeness for expressive power.

Many points in this set of trade-offs are reasonable, so a specification has to allow for both the current set of trade-offs, and also for possible future trade-offs. This means that the specification will not be a complete specification nor even a nearly complete specification.

The approach that has been taken in the specification is to define an expressively powerful description logic, including both a syntax and a semantics, incorporat-

ing those constructs whose meaning has been generally agreed upon by the community. Along with the description logic is a set of interface functions that allow for the construction, manipulation, and querying of description-logic knowledge bases. These functions allow

- the formation of descriptions and sentences;
- the definition of concepts and roles from descriptions;
- the assertion of sentences, including ground facts about individuals and simple rules about concepts;
- the creation of individuals and reasoning about their identity;
- the making of local closed-world statements;
- the making of default statements about instances of concepts;
- the retracting of previously-told assertions; and
- the querying of knowledge bases.

The non-query functions are defined by their effect on an abstract knowledge base, which is a collection of statements in the description logic. The results of the query functions are (mostly) defined by semantic relationships between the knowledge base and the query.

Because it is impossible to efficiently perform inference in the full description logic, conforming systems are not required to completely implement it. Conforming systems are free to recognize only a subset of the syntax of the logic, and need not even perform complete reasoning in the subset that they do recognize. However, such systems must use this logic as the ideal meaning of their knowledge bases, and must perform "sound" reasoning with respect to the logic.

Conforming systems are not completely free in what portion of the logic they choose to address. There is a core portion of the logic that all conforming systems are required to implement; in this way a minimal competence is required for all conforming systems. The core is not just a syntactic subset of the full logic—complete inference on even very minimal subsets of the logic is very difficult—but is instead a set of constructs that must be recognized, along with a set of inferences that must be performed on these constructs.

Most of the debate on the specification has involved the details of this core. The constructs to include in the core, the inferences to perform on them, and how to specify these inferences have all been subjects of debate. This was to be expected, as the specification of the core is where the specification is making decisions on matters that have been decided in different ways by different systems. Devising a core that is both reasonable and non-trivial is an interesting exercise in how to balance various representation and implementation concerns.

There is now (July 1992) a second draft of the complete specification that has been distributed to interested parties. Some changes still need to be made to this draft. First, formal work in description logics has advanced, and should be incorporated into the specification. Second, there are portions of the draft, particularly in the inferences required in the core, that are objectionable to some parties. By September 1992, there should be a third draft prepared and discussed, and by the end of October 1992 a final version of the specification should be available. Also, a method for demonstrating compliance with the specification will be developed.

Future work in the KRSS group effort on description-logic based systems will then consist of augmenting the specification as new formal work on description logic produces relevant results and as new implementation techniques make it possible to extend the core. Also, other families of knowledge representation systems may be given the same treatment, provided that developers are interested.

3 KNOWLEDGE QUERY AND MANIPULATION LANGUAGE (KQML)

The *External Interfaces* working group was originally charged with addressing the general problem of defining standard high-level interfaces for knowledge representation systems. This was seen as including such diverse interfaces as those to other KR systems, DBMSs, active sensors, and human users. Over the past two years, this working group has focused on and experimented with a somewhat narrowed and more focused problem – designing a common high-level language (KQML) and associated protocol which can be used by software systems for the run-time sharing of information and knowledge. This section briefly describes the current status of the effort to specify KQML and experiment with its use in several testbeds.

3.1 OVERVIEW

The Knowledge Query and Manipulation Language (KQML) is both a message format and a message-handling protocol to support run-time knowledge sharing among agents. KQML can be used as a language for an application program to interact with an intelligent system or for two or more intelligent systems to share knowledge in support of cooperative problem solving. KQML focuses on an extensible set of *performatives*, which defines the permissible operations that agents may attempt on each other's knowledge and goal stores. The performatives comprise a substrate on which to develop higher-level models of interagent interaction such as contract nets and negotiation [8, 33].

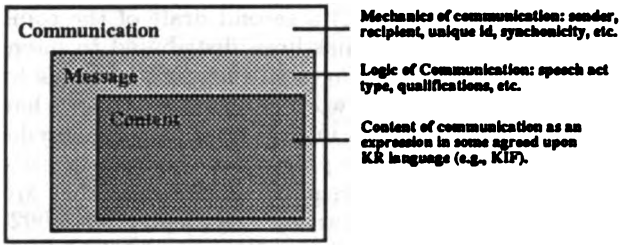


Figure 1: KQML expressions can be thought of as consisting of a content expression encapsulated in a message wrapper which is in turn encapsulated in a communication wrapper.

In addition, KQML provides a basic architecture for knowledge sharing through a special class of agent called *communication facilitators*. These agents coordinate the interactions of other agents by providing such functions as:

- identification of other agents with which to communicate both explicitly via "names" or "addresses" or implicitly via declared topics of interest or capabilities,
- maintaining registration databases of knowledge services offered and sought by agents,
- communication services (e.g., forwarding information from one agent to other interested agents), and
- content translation to bridge semantic and ontologic differences between end agents.

These functions are embodied in special performatives (which take messages as arguments), and in the way that facilitators treat messages received from application agents.

The ideas which underly the evolving design of KQML are currently being explored through experimental prototype systems which are being used to support two testbeds: the Palo Alto Collaborative Testbed (PACT) [9] which is focused in the concurrent engineering domain, and the DARPA/Rome Planning Initiative (DRPI) which deals with military transportation planning [13].

3.2 KQML EXPRESSIONS ARE LAYERED

KQML expressions consist of a content expression encapsulated in a message wrapper which is in turn encapsulated in a communication wrapper, as shown in Figure 1. Thus the language is thought of as being divided into three layers: content, message and communication. The content layer contains an expression in some language which encodes the knowledge to be conveyed. The format of this expression is unimportant to KQML; it can carry any type of content expressed in any representation language which follows some general syntactic constraints (currently, the content expression must be an s-expression). However, there are emerging conventions for knowledge representation (e.g., Interlingua, KIF [17], etc) and standards for *persistent objects* (e.g., the OMG Object Request Broker) which may prove to be very valuable in the near future.

The primary purpose of the message layer is to identify the speech act or performative that the sender attaches to the content, such as an assertion, a query or a command, and any of a small set of qualifiers that may be appropriate to that performative. In addition, since the the content is opaque to KQML, this layer also includes optional features describing the content's language, the ontology it assumes and a descriptor naming a topic within the ontology. These features make it possible for the protocol implementation to analyze, route and properly deliver messages even though their content may be inaccessible.

The final communication level adds a second layer of features to the message which describe the lower level communication parameters, such as the identity of the sender and recipient, a unique identifier associated with the communication and whether the communication is meant to be synchronous or asynchronous. These are used by the network layer which provides reliable transfer of bytes between processes on a network.

3.3 KQML PERFORMATIVES

The message layer is used to encode a message that one application would like to have transmitted to another application and forms the core of the language, determining the kinds of interactions one can have with a KQML-speaking agent. It can be thought of as a "speech act layer", since an important attributes to specify about the content is what kind of "speech act" it represents – an assertion, a query, a response, an error message, etc.

Structure. Conceptually, a KQML message consists of an operator or *performative*, its associated arguments which constitute the real *content* of the message and a set of optional arguments which describe the content in a standard, language-independent manner. For example, a message representing a query about the location of an particular airport might be encoded as:

```
(ask (geoloc lax (?long ?lat))
     :number_answers 1
     :ontology drpi_geo)
```

In this message, the KQML performative is *ask*, the content (i.e., knowledge being sought) is (*geoloclax*(*?long?lat*)), the number of answers requested is 1, the language in which the content is expressed is (by default) *kif* and the ontology to be assumed is that named by the token *drpi*$_g$*eo*. The same

general query could be conveyed in using standard Prolog as the content language in a form that requests the set of all answers as:

```
(ask "geoloc(lax, (Long,Lat))"
     :language standard_Prolog
     :number_answers all
     :ontology drpi_geo)
```

Semantics. It is our intention to allow the set of KQML performatives to be extensible. We will identify a core set of performatives that will have a well defined meaning. An KQML-speaking agent need not implement or handle all of the performatives in this core, but for those it does, it must adhere to the standard semantics. Moreover, it is our goal to provide a standard mechanism by which one can define the semantics of new performatives, allowing the set to be extended. The semantics of the core performatives will be defined in terms of a smaller set of *primitive performatives*. The semantics of these primitive performatives are defined with respect to a simple and general model of agents in which each agent as a store of information structures (i.e., "belief" like items) and a store of goals structures (i.e., items which may effect the agent's future behavior).

Primitive Performatives. We are currently working with a set of four primitive performatives from which we believe the core and various interesting extensions can be defined. These four primitives provide operators to present an agent with items to add (*ADVISE*) and remove (*UNADVISE*) from its information store and to add (*ACHIEVE*) and remove from (*FORGET*) its goal store. These four performatives are primarily used as a means to specify the semantics of the larger core performatives.

Core Performatives. The core set of performatives is expected to include several dozen operators which most KQML-speaking agents will support. If an agent accepts a message with a core performative, it must adhere to its agreed upon semantics. Some of these performatives will accept optional arguments which serve as qualifier. Figure 2 shows some examples of performatives that are in the current specification.

Messaging via Facilitators. Any substantial collection of interacting agents will require some structure on information flow [20, 28, 32]. For this reason, KQML introduces a class of communication facilitator agents that help manage the message traffic among application agents. Facilitator agents can route performatives to appropriate agents (MONITOR performatives in particular), record the performative-processing abilities of new agents, and bridge the capabilities of superficially incompatible agents (through buffering, translation, and problem decomposition). These facilitation functions will be reflected in new core performatives, e.g., (**FORWARD** *agent-name message*) and (**DISTRIBUTE** *message*).

- **(ASSERT P)** - Add P to the agent's information store, performing whatever reasoning the agent can perform.
- **(RETRACT P)** - Remove P from the agent's information store if present, signalling an error if not present and performing whatever reasoning the agent can perform.
- **(ASK P)** - Query the agent's information store to find answers matching query P. The number of answers returned is governed by an optional argument.
- **(GENERATOR P)** - Reply with a *generator* that the recipient can use to elicit a stream of answers to the query P.
- **(MONITOR P)** - Modify the agent's goal store to cause it to inform the sender whenever a sentence matching P becomes true.

Figure 2: These are a few of the core KQML performatives.

Software Architecture. As Figure 3 shows, a typical KQML-speaking agent will be built using two reusable pieces – an interface between the agent's system language (e.g., LOOM or Prolog) which ties communication actions to system actions, and a router which handles the low-level communication chores necessary to talk to other agents. These might all be done within a single process (e.g., in Lisp) or might include several processes (e.g., the router might be done in C or Perl).

3.4 STATUS AND OPEN ISSUES

The ideas which underly the evolving design of KQML are currently being explored through experimental prototype systems which are being used to support

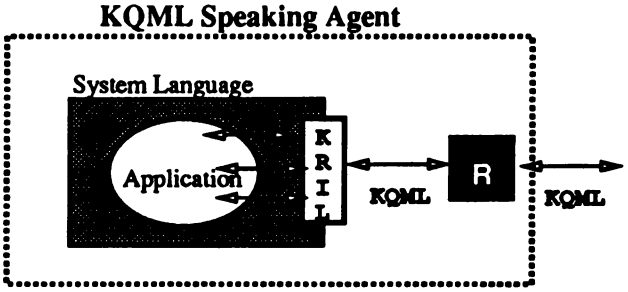


Figure 3: A typical KQML-speaking agent will be built using two reusable pieces – an interface between the agent's system language (e.g., LOOM or Prolog) which ties communication actions to system actions, and a router which handles the low-level communication chores necessary to talk to other agents.

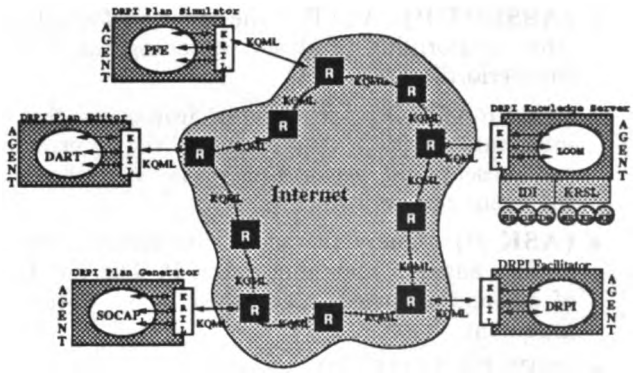


Figure 4: KQML will be used as communication language among the various agents which make up the DRPI testbed. It will be used, for example, to support the interchange of knowledge among the planner, the plan simulator, the plan editor and the DRPI knowledge server which is the repository for the shared ontology and access point for common databases.

two testbeds: the Palo Alto Collaborative Testbed (PACT) [9] which is focused in the concurrent engineering domain, and the DARPA/Rome Planning Initiative (DRPI) which deals with military transportation planning.

KQML use in PACT. The Palo Alto Collaborative Testbed (PACT) uses KQML as its medium for agent interaction in support of concurrent engineering. PACT participants modified several existing knowledge-based engineering systems to speak KQML and thereby exchange design and manufacturing knowledge of mutual interest. (For example, the mechanical modeler sends the controls modeler knowledge regarding the dynamics of the design; the power modeler sends the manufacturing process planner knowledge regarding a motor replacement.) These agents find each other in part through facilitators, which handle message forwarding, content-based routing, and simple format translations.

KQML use in DRPI. The DARPA/Rome Planning Initiative is using KQML as the communication language among the various agents that make up the testbed and feasibility demonstrations. Figure 4 shows KQML being used, for example, to support the interchange of knowledge among the planner, the plan simulator, the plan editor and the DRPI knowledge server, which is the repository for the shared ontology [21] and access point to common databases through the *Intelligent Database Interface* [23, 29]

Open Issues. The design of KQML has continued to evolve as the ideas are explored and feedback is received from the prototypes and the attempts to use them in real testbed situations. We mention here a few of the important issues that we expect to be addressed in the coming year.

The core set of performatives is still undergoing revision as we experiment with its use. This set needs to be stabilized and well specified. In particular, we need to refine the model of what a communication facilitator is and what services it might offer so that we develop a good set of performatives to support their effective use.

A method for defining new extensions to the core set needs to be worked out. This includes a method for defining them for humans as well as a method to allow one agent to define a new performative to another.

The basic model of a knowledge representation agent that we have been working with is quite simple. One of several extensions that may be needed, for example, is a mechanism to define contexts within an agents information and goal stores.

An important part of KQML will be the protocols associated with the different performatives. There are some general issues which go beyond defining the semantics of particular performatives that must be addressed. These general protocols include such things as refusing to accept a message, error reporting, security, and transaction oriented processing.

4 SHARED, REUSABLE KNOWLEDGE BASES

The SRKB Working Group (Shared, Reusable Knowledge Bases) of the DARPA Knowledge effort is working on the problem of sharing the *content* of formally represented knowledge. Sharing content requires more than a formalism (KIF) and communication protocol (KQML). Of course, understanding the nature of what needs to be held in common between communicating agents, or between the author of a book and its reader, is a fundamental question for philosophy and science. The SRKB group is focusing on the practical problem of building knowledge-based software that can be shared and reused as off-the-shelf technology. The charter of the group is to identify the technical barriers to the sharing and reuse of formally represented knowledge by AI programs, and to provide a forum for experimentation with possible approaches.

4.1 STRATEGY: COMMON ONTOLOGIES AS A SHARING MECHANISM

The strategy is to focus on common ontology as the sharing mechanism [27, 18]. What is a common ontology? Every knowledge-based system is based on some conceptualization of the world: those objects, processes, qualities, distinctions, and relationships that matter for performing some task. A program (or its programmer) makes ontological commitments to a conceptualization by embodying these concepts, dis-

tinctions, etc. in a formal representation and using knowledge formulated in that representation during problem solving. By common ontology we mean an explicit specification of a the ontological commitments of a set of programs. Such a specification is an objective description—interpretable outside of the programs—of the concepts and relationships that the programs assume and use when interacting with other programs, knowledge bases, and human users.

Operationally, a common ontology can be specified as a set of definitions of representational terms used to construct expressions in a knowledge base, such as classes, relations, slots, and object constants. To make a common ontology shareable, the definitions should consist of human-readable text and machine-enforceable, declarative constraints (i.e., axioms) on the well-formed use of the terminology. The set of terms in a common ontology need not include all the terms used internally in participating programs. Rather, the shared vocabulary defined in a ontology is used for specifying the coupling between programs and knowledge bases (at design time) and for knowledge-level communication among agents (at run time). We hope to enable large-scale sharing and reuse of knowledge bases and knowledge based systems by making common ontologies available as open specifications, much like interchange formats and communication protocols.

The initial activities of the working group have been to explore the research issues in knowledge sharing, and to identify areas where it might be practical and useful to specify common ontologies. The Summer Ontology Project, held at Stanford in 1990, studied the collaborative, multi-disciplinary development of reusable ontologies for describing electromechanical devices and their designs. One outcome was the observation that several approaches to device modeling, from digital circuit modeling to rigid body dynamics, seemed to make commitments to lumped-element models of physical devices. In a lumped-element model, the behavior of a device is described in terms of values of functions (state variables) that map a single independent variable (e.g., time, but not space) to physical quantities (position, force, etc.). A preliminary ontology was proposed to formalize these concepts.

In March of 1991, the SRKB group met at Pajaro Dunes to characterize some of the research issues. There was some controversy about whether it is premature to "standardize" ontologies of any sort, especially those designed to be comprehensive over tasks and domains. Instead, a series of collaborative, grassroots experiments were proposed, in which two or more research groups identify potential candidates for knowledge sharing.

In the past year, several collaborations have begun, and a set of ad hoc subgroups have been formed to study these ontological niches. Each subgroup is tasked with identifying, collecting, making available, and analyzing ontologies for knowledge sharing. We will describe the efforts of these groups within a framework of models of sharing and reuse.

4.2 MODELS OF KNOWLEDGE SHARING AND REUSE

Three models of sharing and reuse are being explored, and in each, common ontologies play an enabling role.

First is the **library model**, in which bodies of formally represented knowledge are available as off-the-shelf products, like books in a library. In this model, knowledge bases are designed artifacts, and the role of SRKB to help make them available and reusable.

Two ad hoc subgroups are currently active within the library model of sharing. One is an effort by representatives of projects in qualitative physics to specify a common language for model fragments. Model fragments are conceptual building blocks for programs that formulate and assemble engineering models of device behavior, using techniques such as compositional modeling [11]. For example, idealized components such as resistors and physical processes such as liquid flow are represented by model fragments, which are composed to produce simulation models of complete systems. The language under development is a unification of model formulation and simulation systems such as QPE, DME, and QPC, and should enable a community library of model fragments that can be directly executed by these systems. The axiomatic semantics of the language will be expressed in KIF, and the ontological commitments of these programs will be specified as an ontology.

A second subgroup, following up on the Summer Ontology Project, is developing a family of ontologies for specifying various styles of engineering modeling. It is formalizing the classes of algebras used in constraints (e.g., with or without differential equations; qualitative operators), the assumptions underlying component/connection topologies, and the various styles of dynamics analysis (e.g., Newtonian, LaGrangian, Kane's method). This work is complementary to the composition modeling effort; any of these styles of modeling can be formulated using the model fragment language.

A preliminary finding is that the ontological commitments of a given approach to modeling may be factored into separate ontologies. These ontologies form an inclusion hierarchy, where each ontology can inherit (by set inclusion) the definitions of included ontologies. For example, the original proposal for a lumped-element ontology has since been divided into several ontologies, including *continuous-state-space* (commits to describing behavior using state variables) and *hierarchical-component-assembly* (objects are structured into components related by connections and

part-of relations). To specify how state variables are associated with components, one writes a third ontology that includes the other two, adding a few additional constraints. To support this sort of modular partitioning of ontologies, the interlingua committee is considering context mechanisms such as Cyc's microtheories.

A second mode of sharing and reuse under investigation is the **software engineering model**. A standard approach to making software reusable is to decompose complex programs into modular pieces, and to provide a formal specification of the inputs, outputs, and function computed by each piece. Knowledge-based systems are like other software in this respect, except that they operate on a special input called the "background knowledge base" or "domain theory." Reusable modules are designed so that the same code can be used on several knowledge bases. However, to write these knowledge bases the developer needs to understand the ontological assumptions and commitments made in the code. An ontology that defines the vocabulary with which to write the knowledge bases can help determine which software module to use on a given problem, how to provide it the necessary domain knowledge, and whether the knowledge base meets the input requirements of the software.

A significant effort is under way in the knowledge acquisition community to formally characterize the tasks being performed by knowledge based systems, and to design modular problem-solving methods that can be combined to address these tasks [24]. For example, complex, amorphous tasks such as diagnosis and planning have been decomposed into more generic subtasks that can be solved with reusable methods such as simple classification, abductive assembly, and varieties of constraint satisfaction. An subgroup led by Mark Musen is studying ways to describe these tasks and methods, and has begun to define ontologies that specify the input and output assumptions of reusable methods.

A second subgroup, headed by Ed Hovy and Doug Skuce, is identifying and analyzing the comprehensive, *top-level* ontologies that are intended to be general across domains and tasks. A motivating application for such ontologies is natural language processing. NLP techniques needs a way to couple to domain knowledge bases (for something to talk about) without committing the programs to particular subject matter areas. For example, the Penman language generation system's "Upper Structure" ontology [3] divides the world up according to the major type distinctions made in English and German (Objects of various types, Processes and Relations of various types, Qualities, etc.). A developer customizes Penman to a particular application domain by defining the domain's concepts as specializations of the appropriate Upper Structure concepts. As a result, the domain concepts inherit the necessary linguistic annotations from their Upper Structure ancestors. In general, such top-level ontologies can be viewed as a software reuse mechanism for programs parameterized by large knowledge bases.

Another subgroup is looking at ontologies that specify semiformal representations of decision making and design rationale (Jeff Bradshaw, Jin Tae Lee, and Charles Petrie). In semiformal rationale support systems, users organize text describing design decisions in to a hypertext document that supports a fixed vocabulary of node types (classes) and link types (relations). For example, in the gIBIS ontology [7], decisions are described in terms of *issues*, *arguments*, and *positions*, and these node types are linked by relations such as *supports* and *objects-to*. The documents structured by these terms are called semiformal or semistructured, since only the node and link types are machine interpretable and the contents of the nodes are not formalized. Several methodologies for developing semiformal documents, and tools to support them, are based on these ontologies of node and link types.

A third kind of sharing and reuse is the **reference model**, typically used to define an integration framework for a family of application programs. A reference model defines the concepts in a domain and/or problem area that are common to the set of application tasks. For example, a reference model for digital circuits includes a formalism for describing the netlist, which is a representation of circuit topology. The reference model ontology commits the participating tools to the existence of shared objects such as components connected by ports in a netlist; this is necessary to enable tools to exchange data.

An international standards effort called PDES/STEP is working on a family of reference-model ontologies for product data, starting by defining primitives for geometry and working toward high level descriptions of behavior and functionality. The DARPA knowledge sharing effort is exploring avenues for collaboration with the PDES organization.

Within the SRKB working group, ad hoc subgroups are studying reference-model ontologies for user interface toolkits (Jim Foley and Bob Neches), manufacturing enterprise models (Mark Fox), and planning/scheduling (Don McKay, Masahiro Hori).

4.3 TECHNICAL SUPPORT FOR ONTOLOGIES – ONTOLINGUA

Each of the subgroups of the SRKB are charged with identifying and collecting ontologies, and making them available in a form amenable to analysis and possible reuse. However, existing ontologies are either incompletely formalized or written in a specific knowledge representation tool. To address this problem, a system called Ontolingua has been developed [19]. Ontolingua

is a mechanism for defining ontologies *portably*, that is, independent of specific representation systems. It allows the definition of classes, relations, and distinguished objects using KIF sentences, and translates these definitions into several implemented representation systems.

Ontolingua's design demonstrates the use of a common ontology to facilitate sharing and reuse (in this case, of ontologies). Translation from a very expressive language (KIF) into restricted languages is inherently incomplete. Therefore, Ontolingua supports a subset of legal sentences that can be translated into a class of commonly-used representation systems: the object-centered or frame-based systems. These implemented systems commit to particular ways of organizing and specifying knowledge about objects, such as inheritance hierarchies and slot descriptions. These ontological commitments are captured in the Frame Ontology, which defines a vocabulary for describing classes, binary relations, and second-order relationships among them (e.g., subclass, instance, class partitions, slot-value restrictions). Ontolingua recognizes the use of Frame Ontology concepts in KIF sentences, and translates them into the special syntax of each target representation system. The Frame Ontology, on top of a syntactically restricted KIF, defines a language for portable ontologies. The Ontolingua software operationalizes the language by providing automatic translation into implemented representation systems.

5 SUMMARY

Moving beyond the capabilities of current knowledge-based systems will require development of knowledge bases that are substantially larger than those we have today. It will require knowledge-based systems to communicate with other knowledge-based systems and conventional software systems in carrying out their functions. Meeting these challenges on a broad scale will require development new knowledge-sharing technology and shared conventions. The on-going efforts in the Knowledge Sharing Effort represent steps in this directions. The efforts underway are neither complete nor comprehensive – they represent an initial first steps that will result in valuable experience and understanding, will identify shortcomings in current methods and point to new research directions, will encourage others to focus on solving problems encountered in knowledge sharing, to explore alternatives and to enhance the state of the art.

Acknowledgments

This effort is supported by NSF grant IRI-9006923, DARPA/NASA-Ames contract NCC 2-719, and a cooperative agreement between USC/ISI and the Corporation for National Research Initiative. We would also like to acknowledge members of the Knowledge Sharing Effort, too numerous to name individually.

References

[1] Franz Baader, Hans-Jürgen Bürckert, Jochen Heinsohn, Bernhard Hollunder, Jürgen Müller, Bernhard Nebel, Werner Nutt, and Hans-Jürgen Profitlich. Terminological knowledge representation: A proposal for a terminological logic. A DFKI note., June 1991.

[2] Franz Baader and Bernhard Hollunder. KRIS: Knowledge Representation and Inference System—system description. Technical Memo TM-90-13, Deutsches Forschungszentrum für Künstliche Intelligenz, November 1990.

[3] John A. Bateman. Upper modeling: A general organization of knowledge for natural language processing. Penman development note, USC/Information Sciences Institute, 1989.

[4] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, Portland, Oregon, 1989.

[5] Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 34–37, Austin, Texas, August 1984. American Association for Artificial Intelligence.

[6] Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, Lori Alperin Resnick, and Alex Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In Sowa [34], pages 401–456.

[7] Jeff Conklin and M. L. Begeman. gIBIS: A hypertext tool for exploratory policy discussion. In *Proceedings of the 1988 Conference on Computer Supported Cooperative Work (CSCW-88)*, pages 140–152, Portland, Oregon, 1988. ACM.

[8] Susan E. Conry, Robert A. Meyer, and Victor R. Lesser. Multistage negotiation in distributed planning. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 367–384. Morgan Kaufman, 1988.

[9] Mark Cutkosky, Robert Engelmore, Richard Fikes, Thomas Gruber, Micheal Genesereth, William Mark, Jay Tenenbaum, and Jay Weber. Pact: An experiment in integrating concurrent engineering systems. *IEEE Computer*, 1992. To appear in a special issue on computer-supported concurrent engineering.

[10] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. In *Proceedings of the Second In-*

ternational Conference on Principles of Knowledge Representation and Reasoning, pages 151–162. Morgan Kaufmann, May 1991.

[11] Brian Falkenhainer and Ken Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 51:95–143, 1991.

[12] Richard Fikes, Mark Cutkosky, Tom Gruber, and Jeffrey Van Baalen. Knowledge sharing technology project overview. Technical Report KSL 91-71, Stanford University, Knowledge Systems Laboratory, 1991.

[13] T. Finin, R. Fritzson, and D. McKay et. al. A language and protocol to support intelligent agent interoperability. In *Proceedings of the CE & CALS Washington '92 Conference*, June 1992.

[14] T. Finin, R. Fritzson, and D. McKay et. al. An overview of KQML: A knowledge query and manipulation language. Technical report, Department of Computer Science, University of Maryland Baltimore County, 1992.

[15] Michael R. Genesereth. Designworld. In *Proceedings of the 1991 International Conference on Robotiocs and Automation*, pages 2785–2788, 1991.

[16] Michael R. Genesereth. Knowledge interchange format. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Proceedings of the Conference of the Principles of Knowledge Representation and Reasoning*, pages 599–600. Morgan Kaufmann Publishers, Inc., 1991.

[17] Michael R. Genesereth, Richard E. Fikes, and et al. Knowledge interchange format, version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.

[18] Thomas R. Gruber. The role of common ontology in achieving sharable, reusable knowledge bases. In J. A. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 601–602, Cambridge, MA, 1991. Morgan Kaufmann.

[19] Thomas R. Gruber. Ontolingua: A mechanism to support portable ontologies. Technical Report KSL 91-66, Stanford University, Knowledge Systems Laboratory, 1992. June 1992 Revision.

[20] Michael N. Huhns, David M. Bridgeland, and Natraj V. Arni. A DAI communication aide. Technical Report ACT-RA-317-90, Microelectronics and Computer Technology Corporation, Microelectronics and Computer Technology Corporation, 3500 West Balcones Center Drive, Austin TX 78759-6509, October 1990.

[21] Nancy Lehrer. DARPA/Rolm Laboratory Planning and Scheduling Initiative, Knowledge Representation Specification Language: KRSL Version 2.0. Language specification and manual, 1992.

[22] Robert MacGregor. Loom users manual. Working Paper ISI/WP-22, USC/Information Sciences Institute, 1990.

[23] Don McKay, Tim Finin, and Anthony O'Hare. The intelligent database interface. In *Proceedings of the 7th National Conference on Artificial Intelligence*, August 1990.

[24] Mark A. Musen. Overcoming the limitations of role-limiting methods. *Knowledge Acquisition*, 4(2):165–170, 1992.

[25] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2):235–249, May 1990.

[26] Bernhard Nebel. Terminological cycles: Semantics and computational properties. In Sowa [34].

[27] Robert Neches, Richard Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Senator, and William R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):16–36, 1991.

[28] Mike P. Papazoglou and Timos K. Sellis. An organizational framework for cooperating intelligent information systems. *International Journal on Intelligent and Cooperative Information Systems*, 1(1), (to appear) 1992.

[29] J. Pastor, D. McKay, and T. Finin. Viewconcepts: Knowledge-based access to databases. In *Proceedings of the First International Conference on Information and Knowledgement*, November 1992.

[30] Peter F. Patel-Schneider. Undecidability of subsumption in NIKL. *Artificial Intelligence*, 39(2):263–272, June 1989.

[31] Christof Peltason, Albrecht Schmiedel, Carsten Kindermann, and Joachim Quantz. The BACK system revisited. KIT-Report 75, Department of Computer Science, Technische Universität Berlin, September 1989.

[32] Kirk Sayre and Michael A. Gray. Backtalk: A generalized dynamic communication system for DAI. Technical Report CSIS-91-004, The American University, Washington DC, August 1991.

[33] Sandip Sen and Edmund H. Durfee. A formal study of distributed meeting scheduling: preliminary results. In *Proceedings of the ACM Conference on Organizational Computing Systems*, pages 55–68, November 1991.

[34] John Sowa, editor. *Principles of Semantic Networks: Explorations in the representation of knowledge*. Morgan-Kaufmann, San Mateo, California, 1991.

Twelve Years of Nonmonotonic Reasoning Research: Where (and What) Is the Beef?

Raymond Reiter
Department of Computer Science
University of Toronto
Toronto, Ontario
M5S 1A4
Canada

It has been twelve years since the publication of the AIJ Special Issue on Nonmonotonic Reasoning. This event had two important consequences: it focussed the attention of the research community at large on the phenomenon of nonmonotonicity in AI, and it marked the beginning of a KR subculture devoted to exploring the applications and formal properties of these reasoning patterns. Today, this subculture is flourishing, with its own workshops, and a technically sophisticated literature, much of which, alas, has become inaccessible to the average AI practitioner (and not a few KR specialists as well!). No apologies are necessary for this state of affairs; it simply means that there are many interesting conceptual and mathematical problems here. In any event, this situation is no different than in other branches of computer science which, like AI, have a strong engineering component, for example databases or programming languages. Nevertheless, the remoteness of this literature from the typical AI system-builder (together, no doubt, with a more general ambivalence within AI about the relevance of logic or even, these days, of symbolic processing) has led many researchers to wonder what on earth this stuff is good for. So maybe it is time to step back and try to assess exactly what has been accomplished during this twelve year period.

Perhaps not surprisingly, I am more optimistic than most. In support of my rosier outlook, my talk will provide examples where I think that nonmonotonic theorizing has provided important insights about, and solutions to, many outstanding problems, not only in AI but in computer science in general. Among these are:

- Semantics for logic programs.
- Logic programming implementations of nonmonotonic reasoning.
- Solutions to the frame and ramification problems in the situation calculus and their relevance to database update transactions, program verification and software specification.
- The remarkable convergence of probabilistic and nonmonotonic formalisms for default reasoning.
- Abduction and diagnostic reasoning.
- Foundations for truth maintenance systems.
- Nonmonotonicity and theories of belief revision.
- Introspection and reasoning about belief.
- Educating the next generation of AI students.

One of the real tests of theorizing in any science is the extent to which it provides a unifying perspective on what had previously been seen as a wide variety of disparate phenomena. Reasoning about blocks and about flying birds involve the same nonmonotonic principles. We should celebrate this insight, and the remarkable fact that we have some ideas about how it can be formally described.

Author Index

PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING: PROCEEDINGS OF THE THIRD INTERNATIONAL CONFERENCE

Edited by Bernhard Nebel (German Research Center for Artificial Intelligence), Charles Rich (Mitsubishi Electric Research Laboratories), and William Swartout (University of Southern California/Information Sciences Institute)

The idea of explicit representations of knowledge, manipulated by general-purpose inference algorithms, underlies much work in artificial intelligence, from natural language processing to expert systems. The KR conferences have established themselves as the leading forum for timely, in-depth presentation of new results in knowledge representation and reasoning.

The papers in this volume, which are twice as long as papers in the general AI conferences, have passed a stringent review process. The topics covered include nonmonotonic logic, taxonomic logic, specialized algorithms for temporal, spatial, and numerical reasoning, and knowledge representation issues in planning, diagnosis, and natural language processing. Within these topics, treatments range from the abstract specification of algorithms and their computational complexity to the analysis of implemented systems. These proceedings are an essential reference volume for researchers and students interested in a detailed view of this key research area.

Additional Titles of Interest from the Morgan Kaufmann Series in Representation and Reasoning

PLANNING AND CONTROL, Thomas L. Dean and Michael P. Wellman

REPRESENTATIONS OF COMMONSENSE KNOWLEDGE, Ernest Davis

PRINCIPLES OF SEMANTIC NETWORKS: EXPLORATIONS IN THE REPRESENTATION OF KNOWLEDGE, edited by John Sowa

REASONING ABOUT PLANS, James F. Allen, Henry A. Kautz, Richard N. Pelavin, and Josh D. Tenenberg

READINGS IN PLANNING, edited by James Allen, James Hendler, and Austin Tate

PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING: PROCEEDINGS OF THE SECOND INTERNATIONAL CONFERENCE (KR '91), edited by James Allen, Richard Fikes, and Erik Sandewall

ISBN 1-55860-262-3
ISSN 1046-9567
Artificial Intelligence

Morgan Kaufmann Publishers
2929 Campus Drive, Suite 260
San Mateo, CA 94403