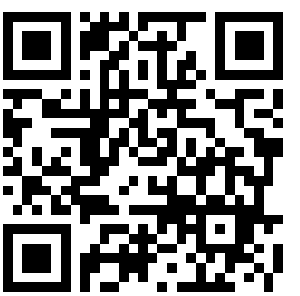


Proceedings of the First International Conference on

Principles of Knowledge Representation and Reasoning

THE MORGAN KAUFMANN

Series in Representation and Reasoning

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

BOOKS

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

Ernest Davis
Representations of Commonsense Knowledge (1989)

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

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

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

PROCEEDINGS

Proceedings of the First International Conference on Knowledge Representation and Reasoning
edited by Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter (1989)

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 about Knowledge: Proceedings of the 1986 Conference
edited by Joseph P. Halpern (1986)

Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge
edited by Moshe Y. Vardi (1988)

Proceedings of the First International Conference on

Principles of Knowledge Representation and Reasoning

Edited by

Ronald J. Brachman
(AT&T Bell Laboratories)

Hector J. Levesque
Raymond Reiter
(University of Toronto)

MORGAN KAUFMANN PUBLISHERS, INC.
SAN MATEO, CALIFORNIA

Editor *Bruce M. Spatz*
Coordinating Editor *Beverly Kennon-Kelley*
Production Manager *Shirley Jowell*
Production Assistant *Elizabeth Myhr*
Cover Designer *Pat Lemmon*
KR'89 Logo Design *Kathryn Finter*
Compositor *Kennon-Kelley Graphic Design*

Library of Congress Cataloging-in-Publication Data

International Conference on Principles of Knowledge Representation and Reasoning (1st : 1989 : Toronto, Ont.)
Proceedings of the First International Conference on Principles of Representation and Reasoning / edited by Ronald J. Brachman, Hector J. Levesque & Raymond Reiter.
p. cm.
ISBN 1-55860-032-9
1. Reasoning--Congresses. 2. Representation (Philosophy)-Congresses. I. Brachman, Ronald J., 1949- . II. Levesque, Hector J., 1951– . III. Reiter, Raymond. IV. Title.
BC177.I57 1989
160–dc19 89-2412
CIP

ISBN 0-55860-032-9
MORGAN KAUFMANN PUBLISHERS, INC.
2929 Campus Drive
San Mateo, CA 94403

93 92 91 90 89 5 4 3 2 1

Acknowledgements

KR'89 would not have been possible without the extensive efforts of a great number of dedicated people.

First, and most important, is our outstanding program committee, who were asked to contribute extraordinary effort in reviewing and comparing an inordinate number of papers, and who did a first-rate job:

James Allen
University of Rochester

Giuseppe Attardi
DELPHI SpA, Italy

Woody Bledsoe
University of Texas

Alan Bundy
Edinburgh University

Eugene Charniak
Brown University

Veronica Dahl
Simon Fraser University

Johan de Kleer
Xerox Palo Alto Research Center

David Etherington
AT&T Bell Laboratories

Koichi Furukawa
ICOT, Tokyo

Hervé Gallaire
ECRC, Munich

Michael Genesereth
Stanford University

Michael Georgeff
The Australian Artificial Intelligence Institute

Pat Hayes
Xerox Palo Alto Research Center

Geoff Hinton
University of Toronto

David Kirsh
MIT

Robert Kowalski
Imperial College of Science and Technology, London

Vladimir Lifschitz
Stanford University

Alan Mackworth
University of British Columbia

Drew McDermott
Yale University

Tom Mitchell
Carnegie Mellon University

Robert Moore
SRI International

Judea Pearl
UCLA

Stan Rosenschein
Teleos Research

Stuart Shapiro
SUNY at Buffalo

Yoav Shoham
Stanford University

William Woods
ON Technology Inc.

A special acknowledgement is also due to David Etherington, David Kirsh, and Yoav Shoham for organizing and moderating our three symposia. Each spent a great deal of time and effort in

creating the idea for his symposium, contacting the participants and cajoling them into appearing, and in bringing the event together.

We would also like to thank a number of reviewers who assisted the program committee in their efforts to evaluate close to 300 papers:

Jun Arima
K. R. Apt
Bruce W. Ballard
Alex Borgida
R. S. Boyer
Mukesh Dalal
Jim Delgrande
Jim des Rivieres
Michael A. Gray
Russ Greiner
Robert F. Hadley
Angela Kennedy Hickman
Larry M. Hines
Michael N. Huhns
Katsumi Inoue
Mitsuru Ishizuka
Matt Kaufmann
Henry Kautz
Kaname Kobayashi
Shigenoba Kobayashi
Gerhard Lakemeyer
Hitoshi Matsubara
Raymond Mooney
Shinichi Morishita
Yasuo Nagai
Steve Nowlan
Joe Nunes
Peter Patel-Schneider
Don Perlis
Charles Petrie
Tony Plato
Dave Plummer
Anand S. Rao
Chiaki Sakama
Kiyokazu Sakane
Ken Satoh
Bart Selman
Hirokazu Taki
Manuela Veloso
Jay Weber
David E. Wilkins

There are a number of other people without whose help we would never have gotten the conference off of the ground. First, and most especially, our gratitude goes to Helen Surridge, for acting as a conference secretary. Helen received and organized all of the submissions to the conference, handled endless phone calls, and generally kept us afloat. We could not have done it without her help. Also, great thanks goes to Wendy Walker, our conference organizer, and also to Carol Plathan and Marina Haloulos. Kathryn Finter designed the KR logo. Peter Patel-Schneider provided the LaTeX macros for the conference papers, and helped immeasurably with other organizational matters. The support of Morgan Kaufmann in the production of these Proceedings was also a crucial ingredient. Thanks go to Mike Morgan, and especially Shirley Jowell for handling the production. Finally (but by no means least) we could not have survived without the patience and support of Gwen Brachman and Pat Levesque.

We would also like to acknowledge the generous support of the organizations that sponsored KR'89: The Canadian Society for Computational Studies of Intelligence (CSCSI — our primary sponsor), The American Association for Artificial Intelligence (AAAI), The International Joint Conferences on Artificial Intelligence (IJCAI), The Canadian Institute for Advanced Research (CIAR), and the Information Technology Research Centre of Ontario. We thank also AISB and ACM SIGART for their cooperation.

Preface

The idea of explicit representations of knowledge, manipulated by general-purpose inference algorithms, dates back at least to the philosopher Leibniz, who envisioned a calculus of propositions that would exceed in its scope and power the differential calculus he had developed. But only very recently has it been possible to design automated systems that perform tasks in part by reasoning in this long-imagined way over a body of represented knowledge. And despite the success of many such systems, the enormous potential for true generality, flexibility, and adaptability is still very far from being realized. Thus, in addition to actually building "knowledge-based systems" of various kinds, a growing number of researchers have become interested in understanding precisely the nature and limitations of these systems.

KR'89 is the first express attempt at bringing together researchers interested in the principles of knowledge representation and reasoning that govern knowledge-based systems. Although the general AI conferences will continue to provide an important venue, it was our belief that this research area had become large enough to warrant an independent meeting where the highest quality technical work that was sometimes too specialized for a general AI audience could be presented and discussed. Moreover, to encourage informal interactions that are sometimes difficult in large crowds, we also wanted to limit the size of the conference to that of the general AI conferences of a decade ago. And, by limiting the size of the program, we could avoid the problems engendered by too many parallel sessions, and could provide lengthy and detailed papers in our Proceedings. These Proceedings represent the written record of this ambitious undertaking.

The response to the KR'89 call for papers was very encouraging, but daunting: we received over 275 submissions, from 27 countries. Each submission was carefully read by at least two members of our program committee. We were pleased to see a remarkable amount of consensus in the reviews. But because of our size constraints, the final selection of papers had to be very competitive. Some papers ended up being excluded not because there was anything specifically wrong with them, but simply because there were too many other submissions that appeared to be better. In the end, we managed to limit the program to the 49 outstanding papers contained in this volume. These papers represent, in our opinion, the very best work being done in the area of knowledge representation and reasoning.

While each of the papers to be presented here in Toronto should be excellent, we thought it worthwhile to go one step further and award a prize for the best paper. However, because it was important to review the final papers for this award, we are unable to include the name of the winner in the Proceedings itself. The winner will be announced at the conference.

As you glance through the conference schedule, you will notice an extra attraction beyond the accepted papers. For this initial conference, we thought it would be interesting and worthwhile to include some non-paper presentations. Thanks to the outstanding efforts of David Etherington, David Kirsh, and Yoav Shoham, we have three mini-symposia to augment our program in the

afternoons. In each case, important and interesting speakers have been invited to present their views on some of the key topics of knowledge representation and reasoning: reasoning about time, nonmonotonic reasoning, and the viability of the entire "knowledge-based" enterprise. We trust that you will enjoy these provocative presentations, and the extensive discussion that has been planned. We are also fortunate to have included here some written material from our invited speakers and discussants.

Finally, while we are hesitant to suggest this (everyone knows what happens to those who suggest something like this), our hope is that KR'89 will be only the first of a series of biennial conferences that highlight outstanding work on the principles of knowledge representation and reasoning. The ultimate success of this and any future KR conferences depends on you, the people who contribute the papers, attend the conference, and read the proceedings. We thank you for your participation, and hope you find the efforts of the organizers and program committee rewarding. KR'91 will depend on your response and contributions.

Ron Brachman	Hector Levesque	Ray Reiter
Program Co-chair	Program Co-chair	Conference Chair

Contents

Presented Papers

A Non-Reified Temporal Logic

Fahiem Bacchus*
Computer Science
University of Waterloo
Waterloo, Ontario
Canada, N2L-3G1
fbacchus@dragon.waterloo.edu

Josh Tenenberg†
Computer Science
University of Rochester
Rochester, New York
U.S.A., 14627
josh@cs.rochester.edu

Johannes A. Koomen‡
Computer Science
University of Rochester
Rochester, New York
U.S.A., 14627
koomen@cs.rochester.edu

Abstract

A temporal logic is presented for reasoning about propositions whose truth values might change as a function of time. The temporal propositions consist of formulae in a sorted first-order logic, with each atomic predicate taking some set of temporal arguments which denote time points, as well as a set of non-temporal arguments. The temporal arguments serve to specify the predicate's dependence on time. By partitioning the terms of the language into two sorts, temporal and non-temporal, time is given a special syntactic and semantic status in the logic without having to resort to reification. The benefits of this logic are that it has a clear semantics and a proof-theory which is easily implemented with standard automated theorem provers. Unlike the first-order logic presented by Shoham, propositions can be expressed and interpreted with respect to any number of temporal arguments, not just with respect to a pair of time points (an interval). We demonstrate the advantages of this flexibility. In addition, nothing is lost by this added flexibility and more standard and useable syntax. To prove this assertion we show that the logic completely subsumes the temporal logic developed by Shoham.

1 Introduction

Many problems in artificial intelligence require reasoning about events or states of the world that have temporal extent. Standard first-order logics have proven useful for reasoning about static propositions and their consequences, but have not been readily adaptable to the greater demands of temporal reasoning. For instance, "block A is on block B" can be represented as ON(A, B), but "block A is on block B from 7 to 12" is less obviously represented. One approach is to add to the predicates additional arguments denoting the temporal interval to which the assertion should be associated: ON(7, 12, A, B). This approach has received little past research attention, being typically abandoned in favor of *reified* logics [Allen, 1984, Lifschitz, 1987, Shoham, 1987] containing truth predicates relating atemporal propositions (e.g., ON(A, B)) to temporal points or intervals (e.g., TRUE[7, 12, ON(A, B)]). In contrast to the recent trends, we demonstrate a logic obtained by including the additional temporal arguments, showing that this preserves the first-order structure of the propositions, has a clear semantics, and a standard proof-theory which, with few augmentations, is easily implemented on the current generation of automated theorem provers.[1] In addition, our logic makes no ontological commitment toward interpreting the temporal objects as either points or intervals, leaving this choice instead to the axiom writer. These advantages are obtained by keeping the logic within a classical first-order framework. We present first the syntax and semantics of our logic, then describe the temporal logic of Shoham, a logic recently presented to deal with problems of preserving the first-order structure of temporally scoped propositions, and finally present a theorem demonstrating the subsump-

*This work was supported by a grant from the Faculty of Mathematics, University of Waterloo.

†This work was supported in part by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under Contract Number F30602-85-C-0008 which supports the Northeast Artificial Intelligence Consortium (NAIC).

‡This work was supported by NSF research grant DCR-8351665.

[1] Our logic can be viewed as being in the same spirit as Green's original work on logic based planning [Green, 1969]. Green used additional state arguments in his predicates, adding the states as extra individuals to the object language. The reified logics on the other hand take the approach of separating the language of states from the language which describes the domain. In order to express the dependence of the domain statements on the current state the formulae of the domain language are reified, *i.e.*, added as extra individuals to the state language. Viewing the time arguments as being state arguments gives the parallel between Green's approach and ours.

tion of Shoham's logic by ours.

2 A Non-Reified Temporal Logic

In the logic that we present, propositions are associated with time objects by including temporal arguments to the functions and predicates. For example, one can represent the assertion "the President of the USA in 1962 died in 1963" as DIED(1963, PRESIDENT(1962, USA)). Temporal objects are distinguished from non-temporal objects by partitioning both the universe of discourse and the symbols of the language used to denote the universe. One can thus specify, for each function and predicate symbol, some number, n, of temporal arguments and some number, m, of non-temporal arguments, and for each function symbol, whether it evaluates to a temporal or non-temporal object.

2.1 Syntax

Our logic, which we will refer to as "BTK," is a standard many-sorted logic having two disjoint sorts, for temporal and non-temporal objects. It is therefore an element of Wang's 2-sorted logical system T_2 [Wang, 1952]. We briefly review the syntax of a two-sorted logic.

The variables, $\mathbf{V}$, are of two different sorts, $\mathbf{V_t}$, and $\mathbf{V_u}$, and for every pair of natural numbers n and m there is a set of (n,m)-ary function symbols, $\mathbf{F}^{(n,m)}$, and a non-empty set of (n,m)-ary predicate symbols, $\mathbf{P}^{(n,m)}$. For both function and predicate symbols the first n arguments are temporal while the last m are non-temporal. We take the constants, $\mathbf{C}$, to be 0-*ary* function symbols. The sort of a function is determined by the sort that the function returns. Hence, the constants are sorted as well.

Terms and wffs are defined in the standard fashion, with the only restriction being that arguments of the correct sort must be given for each function and predicate. We will use "t" to denote temporal terms, and "c" to denote non-temporal terms, both possibly with subscripts. The sort of a term is determined by the sort of its outermost symbol. In addition, we will call predicates that take only temporal arguments *temporal predicates*, and predicates that take only non-temporal arguments *non-temporal predicates*.

A set of inference rules is provided by Wang, which includes universal generalization and modus ponens. For our present purposes we need not include them here, but they can be found in [Wang, 1952]. A BTK language along with the inference rules and proper axioms is a BTK *system*.

2.2 Semantics

A model is defined to be the tuple $\mathcal{M} = \langle\langle T, U\rangle, \sigma\rangle$. T and U are non-empty universes, and σ is an interpretation function that maps each (n,m)-ary temporal function to an (n,m)-ary function from $T^n \times U^m$ to T, each (n,m)-ary non-temporal function to an (n,m)-ary function from $T^n \times U^m$ to U, and each (n,m)-ary predicate to an (n,m)-ary predicate on $T^n \times U^m$. Meaning is assigned to the formulae under the standard interpretation of the truth-functional connectives and quantifiers, except that each quantified variable ranges only over the appropriate universe. We denote the interpretation of ϕ under σ by ϕ^σ.

3 Relativization, Proof Theory and Automated Deduction in BTK

Rather than using a 2-sorted logic for BTK, we could instead have used a standard (one-sorted) logic. Thus, for every BTK system we could have a corresponding BTK′ system, where there is only a single universe, and thus only a single sort for the variables and functions. In addition, the one-place predicates *Temporal* and *Non-Temporal* are part of every BTK′ language. The BTK′ system is then defined analogously to that of BTK, with the addition of the following theorems:

1. $\exists x, y.\ \mathit{Temporal}(x) \wedge \mathit{Non\text{-}Temporal}(y)$
2. $\forall x.\ \mathit{Temporal}(x) \oplus \mathit{Non\text{-}Temporal}(x)$,

where $\oplus$ is exclusive-or. A statement ϕ in BTK can be "relativised" to a statement ϕ' in BTK′, by substituting simultaneously in ϕ, for each expression of the form $\forall x.\alpha$, where x is a temporal variable, an expression of the form

$$\forall x.\ \mathit{Temporal}(x) \rightarrow \alpha,$$

and for each expression of the form $\forall z.\alpha$, where z is a non-temporal variable, an expression of the form

$$\forall z.\mathit{Non\text{-}Temporal}(z) \rightarrow \alpha.^2$$

We then get the following result trivially from Wang, (attributed to Herbrand [Herbrand, 1930]):

> A statement of any system BTK is provable in BTK if and only if its relativisation in the corresponding system BTK′ is provable in BTK′.

Therefore, since BTK′ is a standard first-order system any first-order proof theory can be trivially used as a proof theory for BTK: one only has to relativise every statement of a given BTK system and do deduction in the first order BTK′. In addition, by relativising a BTK system in this fashion, one can automate deduction by using standard automated theorem proving techniques.

It should be noted, however, that one need not relativise the logic in order to obtain either a proof theory or an automated theorem prover for a sorted logic. This is because both are provided by Walther [Walther, 1987], for a sorted clause form logic, based

[2] We are taking existentially quantified variables as defined from universally quantified variables

upon resolution and paramodulation. In fact, Walther gives some strong arguments to indicate that reasoning directly with the sorted logic would be far more efficient. It is a trivial exercise to cast BTK as a variant of Walther's clause form sorted logic and to use his automated reasoner.

Halpern and Shoham [Halpern and Shoham, 1986] have demonstrated that for modal temporal logics the complexity of reasoning is highly dependent on the nature of the temporal domain. A similar situation holds for BTK.

It is well known that the set of valid formulae for first-order structures is recursively enumerable, and many different complete proof theories exist (e.g., the different ones given in [Barwise, 1977][3]). However, these results presuppose the ability to completely axiomatise the properties of the functions and relations defined over the domain. This may not be possible if one requires the domain to have some special structure. For example, if one requires that the temporal domain T be the set of integers, then it is well known that there are no complete axiomatizations of the properties and functions of the integers. In other words, if one places no restrictions on the set of legal BTK models, in particular, if one places no requirements on the structure of the temporal domain, then complete proof theories can be provided for BTK, by the above relativization result and the existence of complete proof theories for first-order logic, or by the use of complete proof theories for sorted first-order logic, like Walther's. On the other hand if one restricts the set of legal BTK models to be models where the temporal domain T has some special structure one cannot necessarily guarantee a complete proof theory: even the relativised first-order BTK′ will not have a complete proof theory.

Although temporal structures like the integers cannot be axiomatized, there are many other temporal structures that can be. These include temporal domains that are linearly ordered (*i.e.*, we require a linear order, but no other structure), models of Peano arithmetic (*i.e.*, we allow non-standard models of the integers), and totally ordered fields. This last is particularly useful. The reals are an instance of a totally ordered field. Hence, if we choose such a temporal structure we will be able to axiomatize its behavior and be assured that all deductions carried out with this axiomatisation will be sound with respect to the reals. Furthermore, it is well known that every totally ordered field has a subfield which is isomorphic to the rationals. This means that we can include in our language temporal constants representing any rational time point. When one considers the fact that our computers can only represent rationals (and only a finite set of rationals at that), it should be clear that one can capture a great deal of useful reasoning about rational time points axiomatically. Another interesting type of temporal domain which has a complete axiomatization occurs when the primitive temporal objects are intervals [Ladkin, 1987]. This choice is perfectly compatible with our definition of BTK.

If the temporal domain, T, of BTK is defined to be any one of these temporal structures, or any other structure for which we have a complete axiomatization, a complete proof theory can be easily generated. One just adds the axiomatisation of the temporal domain to the axiomatization of first-order logic. The first-order rules of inference will provide a complete proof theory when they operate on the union of the temporal and first-order axioms. This can be done in either the sorted context or, via relativisation of the temporal axioms, in the unsorted context.

An argument made by Shoham [Shoham, 1987] is that a non-reified logic such as BTK is insufficient for some of the demands of temporal reasoning:

> This option is not acceptable from our standpoint, although there is nothing technically wrong with it. The problem is that if time is represented as an argument (or several arguments) to predicates, there is nothing general you can say about the temporal aspect of assertions. For example, you cannot say that "effects cannot precede their causes"; at most you can say that about specific causes and effects. Indeed, this first option accords no special status to time—neither conceptual nor notational—which goes against the very spirit of our enterprise.

We deal with these objections by showing that BTK subsumes the logic developed by Shoham (to be referred to as "STL"). Given this result, it is the case that STL can represent the sentence "effects cannot precede their causes" only if BTK can. Thus, STL is no more expressive than a logic obtained by adding additional time arguments to the predicates. In addition, time *is* given a special status in BTK by using a sorted logic that distinguishes temporal objects from all other objects, both semantically (conceptual), and syntactically (notational).

4 Shoham's Logic

In this section we briefly describe STL and discuss the main differences between it and our temporal logic. Shoham's logic is presented in [Shoham, 1987].

STL is sorted in much the same way as BTK. There are a set of temporal constants and variables as well as non-temporal constants and variables. However the treatment of function and relation symbols is different. STL has temporal functions, but these functions

[3] These proof theories give mechanical procedures for generating all valid formulae, thus showing that the set of all valid formulae is recursively enumerable.

can only take temporal arguments—they are a special case of BTK temporal functions, *i.e.*, temporal functions with $m = 0$. Furthermore, STL allows no temporal relations except for the predefined ones '$\leq$' and '='. Non-temporal functions and relations are also treated differently. Syntactically they do not take any temporal arguments, although semantically they are always evaluated with respect to a pair of time *points* (an interval).

The atomic formulae of STL are of two types—formulae formed from the two temporal relations = and $\leq$, e.g., $t_1 = t_2$ or $t_1 \leq t_2$, where t_1, t_2 are both temporal terms,[4] and formulae formed via the "TRUE" construct. Using Shoham's definition,

> If t_a and t_b are temporal terms, $c_1, \ldots, c_m$ are non-temporal terms, and R is a m-ary relation symbol, then
>
> $$\text{TRUE}(t_a, t_b, R(c_1, \ldots, c_m))$$
>
> is an atomic formula.

For example, the sentence "block A is on block B between 7 and 12" would be expressed in STL as

$$\text{TRUE}(7, 12, \text{ON}(\text{A}, \text{B})).$$

"TRUE" is not a relation in STL, nor is it a modal operator; rather, it is a *reifying* context. It asserts that the proposition $R(c_1, \ldots, c_m)$ is true over the interval specified by t_a and t_b. The time points t_a and t_b do not appear as direct arguments to the relation symbol R, nor to any functions which may appear in the c_i's, but they affect the semantic interpretation of these symbols.

The rest of the formulae of STL are built up in the standard manner, by closing off under negation, conjunction and universal quantification. As in BTK, quantification can occur over the time points or over the ordinary individuals, dependent on the sort of variable used.

Semantically STL has, like BTK, a universe of temporal objects and a universe of individuals. Unlike BTK, STL requires that the temporal objects be time *points*. It forces the denotation of $\leq$ to be a connected[5] partial ordering over the time objects, and requires that all of the atomic formulae include two temporal arguments (denoting the starting and ending points of the temporal interval over which the proposition holds). The interpretation function maps the temporal function symbols to functions over the universe of time points. The mapping of the non-temporal function and relation symbols is, however, determined not only by the symbol itself but also by the two time points which occur in the "TRUE" construct.[6] In particular, there is a mapping from non-temporal function symbols and a pair of time points to functions over the universe of individuals. Similarly, there is a mapping from non-temporal relation symbols and a pair of time points to relations over the individuals. Each non-temporal function symbol denotes many different functions over the non-temporal individuals. The particular function that it denotes is determined by the time points in its "TRUE" context, and likewise for non-temporal relation symbols.[7] Once the particular non-temporal function or relation is identified by the time points the rest of the interpretation proceeds in a standard manner. A fuller description of Shoham's logic is provided in the Appendix.

4.1 Comparison of Shoham's Temporal Logic to BTK

There are several implications of Shoham's approach. One is that every non-temporal function and relation is *always* dependent on *exactly* two time points. Thus, for example, there is no way of specifying that a function is dependent on only one time point,

$$\text{LOCATION}(\text{SPACE-SHUTTLE}, t_1),$$

or that a relation is "eternal," *i.e.*, not dependent on time,

$$\text{BLOCK}(A).$$

Since the time dependency is specified semantically the syntax is completely rigid on this matter. Every non-temporal function and relation is *always* dependent on *exactly* two time points. In BTK, there is neither a syntactic commitment to the number of temporal objects that any function or predicate may depend on, nor is there any commitment to interpreting the temporal objects as either intervals or points. It is our position that these choices should not be constrained by the logic, but should be left to the axiom writer to decide.

An additional problem with STL is that there is no simple way of referring to one temporally referenced object within the context of another temporal interval, such as the example "the President of 1962 died in 1963." This is because Shoham requires all non-temporal terms to be evaluated with respect to the *same* temporal terms, *i.e.*, those specified in the TRUE context. To express such a statement in Shoham's logic one has to resort to the more cumbersome use of equality and implication:

$$\forall x[\text{TRUE}(1962, 1963, \text{PRESIDENT}(\text{USA}) = x) \rightarrow \text{TRUE}(1963, 1964, \text{DIED}(x))]$$

[4] Terms in STL are formed in the standard manner, *i.e.*, constants and variables, or functions applied to the proper number of terms. Note, however, that in STL there are no mixed functions, *i.e.*, functions of temporal and non-temporal terms.

[5] Connected means that either $a \leq b$ or $b \leq a$ (or both) for every temporal object a and b.

[6] This is the only place that a non-temporal relation or function can appear.

[7] In this sense, the non-temporal functions can be viewed as *fluents* [McCarthy and Hayes, 1969].

This can be compared with the expression of this statement in BTK given in section 2. It can also be noted that equality is required to express this assertion in Shoham's logic, and it is well known that automated reasoning with equality is very difficult [Wos, 1988]. We will have more to say about reasoning with Shoham's logic below.

A further problem is that Shoham does not allow for temporal predicates, except for the predefined ones $\leq$ and $=$.[8] Thus one would have to extend his formalism to, for instance, embed the MEETS predicate and axioms of [Hayes and Allen, 1987] within STL.

A major difficulty with Shoham's approach is that, since he has chosen to move away from standard (or sorted) first order syntax, first order proof theory, which is purely syntactic, no longer applies. Hence, Shoham's reified logic[9] requires a new proof theory. This means that *one cannot justify the use of Shoham's logic for reasoning about temporal propositions*. There is no reasoning procedure specified that provides any formal guarantees of soundness or completeness.

It may not be very difficult to provide a proof theory for Shoham's logic, but this in itself would not suffice to provide a useful tool for reasoning. One would also have to develop expertise in automating such a proof theory. This may not be an easy task since, as indicated above, there are some examples which force the use of equality which is known to be difficult to automate. The fact that our temporal logic has a standard syntax means that we can take advantage of 20 years of research in automated reasoning.

Our temporal logic is a simple sorted first order logic. It is simple because the sorts do not intersect. Proof theories for sorted first order logics already exist, and are applicable *as is* to our logic. In addition, considerable work has been done on automating such proof theories, [Walther, 1987]. Furthermore, if one chooses to interpret our logic as non-sorted, then standard FOL proof theory applies, as do automated theorem provers for first order logic.

One would hope that there are compensations in using STL in exchange for abandoning standard proof theory. This is, however, not the case. The next section will show that nothing is lost in moving from Shoham's temporal logic to the logic proposed in this paper. It shows that STL is subsumed by our logic in the precise sense that any STL model can be transformed to a BTK model in such a way that there is a one to one correspondence between the sentences satisfied by the STL model and the sentences satisfied by the BTK model. These results also show that there is one way of doing reasoning in STL: *translate it into BTK*.

5 Subsumption of Shoham's Logic

We show that Shoham's logic (STL) is subsumed by the logic proposed in this paper (BTK) by defining two transformations, a syntactic transformation, π_{syn}, and a semantic transformation, π_{sem}.[10] π_{syn} maps sentences of STL to sentences of BTK, while π_{sem} maps models of STL to models of BTK. Using these two transformations we will show that any STL model can be transformed into a BTK model in such a way that the set of sentences satisfied by the BTK model[11] includes the transformed set of STL sentences satisfied by the STL model. In other words, any set of STL sentences can be rewritten as a set of BTK sentences without eliminating any models which satisfy those sentences.

The syntactic transformation is based on a simple idea. In Shoham's logic all predicate symbols and non-temporal function symbols are interpreted with respect to the two time terms which appear as the first two arguments of the "TRUE" construct. In transforming STL to BTK we take these two time terms and add them as explicit temporal arguments to the predicate symbol, and similarly we add them as extra arguments to the non-temporal functions. Temporal functions are unaffected by the transformation, and none of the symbols are altered—they are just rearranged.

The only technical point is that non-temporal terms can be built up from nested application of non-temporal functions. In this case it is necessary to propagate the two temporal arguments recursively to all embedded function terms. For example, the non-temporal term $f(g(h(c)))$ in STL, where f, g, and h are non-temporal functions and c is a non-temporal constant, must be converted to a term of the form $f(t_1, t_2, g(t_1, t_2, h(t_1, t_2, c)))$, where t_1 and t_2 are the propagated temporal terms. Here each of the functions f, g, and h has been converted to functions with two extra temporal arguments.

The following examples should give a good idea of the nature of the syntactic transformation. The STL expressions

1. $\text{TRUE}(t_1, t_2, \text{COLOUR}(\text{HOUSE17}, \text{RED}))$.
2. $\text{TRUE}(t_3, t_4, \text{GENDER}(\text{PRESIDENT}(\text{USA}), \text{MALE}))$.

[8] What we mean here is that the semantic model Shoham defines does not allow for "user defined" temporal relations. He does allow an arbitrary set of temporal functions.

[9] It is perhaps more accurate to refer to Shoham's logic as being *intensional* [Dowty *et al.*, 1981] rather than reified. Shoham does, however, refer to his logic as being a "new reified temporal logic" [Shoham, 1987, Page 103].

[10] Ladkin [Ladkin, 1988] uses a similar approach to map Allen's interval calculus [Allen, 1983] to the language of rational numbers. In doing this he is able to give decision procedures for the interval calculus.

[11] A model, $\mathcal{M}$, satisfies a sentence, α, written $\mathcal{M} \models \alpha$, if $\alpha^{\sigma} = \top$, *i.e.*, if α is true under the interpretation of the model, σ.

3. TRUE$[t_1, f_t(t_1), P(h(g(B)))]$.

will be transformed to the BTK expressions:

1. COLOUR$(t_1, t_2,$ HOUSE17, RED$)$.
2. GENDER$(t_3, t_4,$ PRESIDENT$(t_3, t_4,$ USA$)$, MALE$)$.
3. $P[t_1, f_t(t_1), h(t_1, f_t(t_1), g(t_1, f_1(t_1), B))]$.

The semantic transformation is similar. In STL each non-temporal function or relation symbol actually denotes a set of different functions or relations over the non-temporal individuals. The time points in the "TRUE" context determine which element of the set is picked out for this particular instance. In converting from an STL model to a BTK model we gather up all of the different functions associated with each function symbol and construct a single function which has two extra temporal arguments. The new BTK function has the property that when it is evaluated at a fixed pair of time points it is the same function as the function denoted by the STL symbol when that symbol is interpreted with respect to those time points. The non-temporal relations are transformed in a similar manner.

These transforms are defined formally in the appendix, where we prove the following theorem.

Theorem 1 *Given an STL sentence α and an STL model $\mathcal{M}$ then*

$$\mathcal{M} \models \alpha \quad \textit{iff} \quad \pi_{sem}(\mathcal{M}) \models \pi_{syn}(\alpha).$$

Proof The proof is straight forward, but requires the development of a fair amount of notation. See the appendix for details. ■

This theorem is a formal specification of the manner in which STL is subsumed by BTK, and it has an interesting corollary regarding proof theories.

Corollary 2 *A sound proof theory in BTK can be used to produce sound inferences in STL.*

Proof The proof is extremely simple, given theorem 1. However, to avoid distraction it is given in the appendix. ■

It is natural to ask a similar question about completeness. That is, can a complete proof theory in BTK produce a complete set of inferences in STL. Here, however, the answer is no. If we have that $\alpha \models \beta$ in STL, then we know from theorem 1 that $\pi_{syn}(\alpha)$ entails $\pi_{syn}(\beta)$ in every BTK model which is of the form $\pi_{sem}(\mathcal{M}_S)$, for some STL model, $\mathcal{M}_S$. However these are not the only BTK models, and it is quite possible that in some BTK model which is not a transformed STL model $\pi_{syn}(\alpha)$ is true while $\pi_{syn}(\beta)$ is false. Hence in BTK $\pi_{syn}(\alpha) \vdash \pi_{syn}(\beta)$ would not be sound, even though $\alpha \models \beta$ in STL. Another way of stating this is that there may be a BTK sentence δ which is *not expressible* in STL, and which is consistent with $\pi_{syn}(\alpha)$ and inconsistent with $\pi_{syn}(\beta)$.

6 Translating Shoham's Ontology to BTK

One of the benefits of Shoham's logic is that it does not require the axiom writer to use a fixed ontology of temporally scoped propositions, as, for example, Allen does [Allen, 1984] with his introduction of *properties, events,* and *processes.* Rather, Shoham's logic allows the axiom writer to build her own ontology *axiomatically.*

We argue that Shoham's ontology extends naturally to our logic by virtue of the demonstrated translation, and that, in fact, our ontology is richer, since our logic allows intervals to be the primitive temporal objects rather than being defined by the two endpoints, as in STL. An example showing the translation of the ontology axioms should suffice to demonstrate our claim.

Shoham defines a proposition type x (where proposition types are simply relation symbols with the requisite arguments) to be *downward hereditary* "if whenever it holds over an interval it holds over all of its subintervals." Shoham's axiom schema for this is

$$\begin{aligned}&\forall t_1, t_2, t_3, t_4.\\ &\quad [t_1 \leq t_3 \leq t_4 \leq t_2 \wedge t_1 \neq t_4 \wedge t_3 \neq t_2\\ &\quad \wedge \mathrm{TRUE}(t_1, t_2, x)] \rightarrow \mathrm{TRUE}(t_3, t_4, x).\end{aligned}$$

for all x's of the appropriate type. This translates in BTK to the following schema, for each (n, m)-ary predicate of the appropriate type:

$$\begin{aligned}&\forall t_1, t_2, t_3, t_4.\\ &\quad [t_1 \leq t_3 \leq t_4 \leq t_2 \wedge t_1 \neq t_4 \wedge t_3 \neq t_2\\ &\quad \wedge p(t_1, t_2, c_1, \ldots, c_m)] \rightarrow p(t_3, t_4, c_1, \ldots, c_m).\end{aligned}$$

In addition, the predicate "$\leq$" must be defined axiomatically in BTK, since it is not implicitly defined as it is in STL.[12] In BTK, however, one is not forced to use time *points* so one might alternatively define downward hereditary for a system in which intervals are taken as the interpretation of time objects, as in:

$$\begin{aligned}&\forall i_1, i_2. \mathit{During}(i_2, i_1) \wedge p(i_1, c_1, \ldots, c_m)\\ &\quad \rightarrow p(i_2, c_1, \ldots, c_m),\end{aligned}$$

where it is assumed that *During* has been defined axiomatically.

This same style of translation, then, can be used for any of the other elements of Shoham's ontology: *upward-hereditary, point-downward-hereditary, liquid, gestalt,* etc.

7 Conclusion

A temporal logic has been presented for reasoning about propositions whose truth values might change as a function of time. The temporal propositions consist of formulae in a sorted first-order logic with each

[12] This is because an ordering relation does not make sense for certain temporal structures, e.g., intervals.

atomic predicate taking some set of temporal arguments which denote time objects, as well as a set of non-temporal arguments. The temporal arguments serve to specify the proposition's dependence on time. By partitioning the terms of the language into two sorts, temporal and non-temporal, time is given a special syntactic and semantic status in the logic without having to resort to reification. The benefits of this logic are that it has a clear semantics and a proof-theory which is easily implemented with standard automated theorem provers. Unlike the first-order logic presented by Shoham, propositions can be expressed and interpreted with respect to any number of temporal arguments, not just with respect to a pair of time objects (an interval). In addition, the axiom writer is free to consider the time objects as either points or intervals. By proving that the logic completely subsumes Shoham's, we have demonstrated that nothing is lost by this added flexibility and more standard and useable syntax.

Acknowledgements

Thanks to Andre Trudel and Jay Weber for helpful comments.

A Transformation of STL to BTK

Definition 3 *The syntactic transform, π_{syn}, which maps STL sentences to BTK sentences, is defined recursively as follows. It depends on a syntactic transformation of the non-temporal terms which is defined next.*

1. $\pi_{syn}(t_a \leq t_b) \mapsto t_a \leq t_b$, and $\pi_{syn}(t_a = t_b) \mapsto t_a = t_b$ (*i.e.*, temporal terms and formulae are left intact).
2. $\pi_{syn}\Big(\text{TRUE}(t_a, t_b, p(c_1, \ldots, c_n))\Big) \mapsto$
   $p(t_a, t_b, \pi_{syn}^{[t_a,t_b]}(c_1), \ldots, \pi_{syn}^{[t_a,t_b]}(c_n))$
3. $\pi_{syn}(\neg\alpha) \mapsto \neg\pi_{syn}(\alpha)$
4. $\pi_{syn}(\alpha \wedge \beta) \mapsto \pi_{syn}(\alpha) \wedge \pi_{syn}(\beta)$
5. $\pi_{syn}(\forall x(\alpha)) \mapsto \forall x(\pi_{syn}(\alpha))$, where x can be a variable of either sort.

Definition 4 *The syntactic transform, $\pi_{syn}^{[t_i,t_j]}$, which maps non-temporal terms of STL to terms of BTK is defined as follows.*

1. If c is a non-temporal constant or variable of STL then
   $\pi_{syn}^{[t_i,t_j]}(c) \mapsto c$.
2. $\pi_{syn}^{[t_i,t_j]}(f(c_1, \ldots, c_n)) \mapsto$
   $f(t_i, t_j, \pi_{syn}^{[t_i,t_j]}(c_1), \ldots, \pi_{syn}^{[t_i,t_j]}(c_n))$

The symbols of the corresponding BTK and STL languages are identical, but as is seen from the definition of π_{syn}, non-temporal functions and predicates have two extra temporal arguments.

Next we define the semantic transformation π_{sem}, but in order to do this we first need to provide more detail about the models of STL.

A model of STL is defined to be the tuple

$$\mathcal{M} = \Big\langle TW, \preceq, W, TFN, FN, RL, M \Big\rangle$$

Where:

1. TW is a universe of time points,
2. $\preceq$ is an ordering relation on TW,
3. W is a universe of individuals,
4. TFN is a set of temporal functions, $TW^n \mapsto TW$,
5. FN is a set of non-temporal functions, $W^n \mapsto W$,
6. RL is a set of non-temporal relations in W^n,
7. M is the tuple of interpretation functions $\langle M_1, M_2, M_3, M_4, M_5 \rangle$, where:
   (a) M_1 is a mapping from the time constants to TW,
   (b) M_2 is a mapping from the non-temporal constants to W,
   (c) M_3 is a mapping from the temporal functions to TFN,
   (d) M_4 is a mapping from $TW \times TW \times f \mapsto FN$, where f is the set of non-temporal function symbols,
   (e) M_5 is a mapping from $TW \times TW \times r \mapsto RL$, where r is the set of non-temporal relation symbols.

In the following we denote temporal terms by t_i (for various subscripts i) and non-temporal terms by c_i (note, terms are syntactic entities). We use hatted $\hat{t}$ (usually with subscripts) to denote time points. These are semantic entities which are members of TW, the universe of time points. In addition, we use hatted $\hat{c}$ or $\hat{a}$ (again usually with subscripts) to denote individuals from the semantic domain W.

The meaning of an expression ψ, $M(\psi)$, is defined as follows:

1. If ψ is a temporal variable, then $M(\psi) = VA_t(\psi)$ where VA_t is a variable assignment function over TW.
2. If ψ is a temporal constant, then $M(\psi) = M_1(\psi)$.
3. If ψ is a temporal term of the form $f(t_1, \ldots, t_n)$, then
   $$M(\psi) = M_3(f)(M(t_1), \ldots, M(t_n)).$$
4. If ψ is a non-temporal term, then meaning is assigned to ψ with respect to two time points as follows:
   (a) If ψ is a non-temporal constant, then for all time points $\hat{t_1}$, $\hat{t_2}$,
   $$M(\hat{t_1}, \hat{t_2}, \psi) = M_2(\psi).$$

(b) If ψ is a non-temporal variable, then for all time points $\hat{t_1}$, $\hat{t_2}$,

$$M(\hat{t_1}, \hat{t_2}, \psi) = VA_w(\psi),$$

where VA_w is a variable assignment function over W.

(c) If ψ is a non-temporal function term of the form $f(c_1, \ldots, c_n)$, then for all time points $\hat{t_1}, \hat{t_2}$,

$$M(\hat{t_1}, \hat{t_2}, \psi) = M_4(\hat{t_1}, \hat{t_2}, f)[M(\hat{t_1}, \hat{t_2}, c_1), \ldots, M(\hat{t_1}, \hat{t_2}, c_n)].$$

And finally, a wff ϕ is satisfied under interpretation $\mathcal{M}$ and variable assignment VA, (written $\mathcal{M} \models_{VA} \phi$) as follows:

1. $\mathcal{M} \models_{VA} t_1 = t_2$ iff $M(t_1) = M(t_2)$.
2. $\mathcal{M} \models_{VA} t_1 \leq t_2$ iff $M(t_1) \preceq M(t_2)$.
3. $\mathcal{M} \models_{VA} \text{TRUE}(t_1, t_2, p(c_1, \ldots, c_n))$ iff
$$\langle M[M(t_1), M(t_2), c_1], \ldots, M[M(t_1), M(t_2), c_n]\rangle \in M_5[M(t_1), M(t_2), p].$$

Truth is assigned to non-atomic formulae in the standard fashion. Note that predicates are interpreted with respect to two time points, just as were the non-temporal functions.

We now define the semantic transformation of an STL model.

Definition 5 *The semantic transformation of $\mathcal{M}$, $\pi_{sem}(\mathcal{M})$, is a BTK model constructed as follows.*

1. $T = TW$, the universe of time points is the same.
2. $U = W$, the universe of individuals is the same.
3. If f is a temporal function symbol of STL, then $f^\sigma = M_3(f)$.
4. If f is an n-ary non-temporal function symbol of STL, then the following set of $n+3$ ordered tuples is the interpretation of f under $\pi_{sem}(\mathcal{M})$:
$$f^\sigma = \{\langle \hat{t_1}, \hat{t_2}, \hat{c}_1, \ldots, \hat{c}_n, \hat{a}\rangle | \hat{t_1}, \hat{t_2} \in TW \text{ and } (M_4(\hat{t_1}, \hat{t_2}, f))(\hat{c}_1, \ldots, \hat{c}_n) = \hat{a}\}.$$
Note that this set does in fact define a function. Given any tuple $\langle \hat{t_1}, \hat{t_2}, \hat{c}_1, \ldots, \hat{c}_n\rangle$, M_4 maps f, $\hat{t_1}$, and $\hat{t_2}$ to a unique function over W^n $(= U^n)$. Hence, the $\hat{c_i}$'s will then map to a unique element $\hat{a}$ of W $(= U)$.
5. If P is a predicate symbol of STL, then
   a) if P is $\leq$, then $P^\sigma = \preceq$, *i.e.*, the semantic ordering relation on TW;
   b) if P is $=$, then $P^\sigma = \{\langle \hat{t}, \hat{t}\rangle | \hat{t} \in T\}$;
   c) if P is an n-ary non-temporal predicate symbol of STL then the following set of $n+2$-ary tuples is the interpretation of P under $\pi_{sem}(\mathcal{M})$:
$$P^\sigma = \{\langle \hat{t_1}, \hat{t_2}, \hat{c}_1, \ldots, \hat{c}_n\rangle | \hat{t_1}, \hat{t_2} \in TW \text{ and } \langle \hat{c}_1, \ldots, \hat{c}_n\rangle \in M_5(\hat{t_1}, \hat{t_2}, P).\}$$
6. To complete the definition of σ, we choose an arbitrary mapping of the temporal variables to T and an arbitrary mapping of the non-temporal variables to U. Finally, we maintain the STK denotations of all constants, *i.e.*, $t^\sigma = M_1(t)$ for all time constant symbols t, and $c^\sigma = M_2(c)$ for all non-temporal constant symbols c.

Now we can prove the main technical result.

Theorem 1 *Given an STL sentence α and an STL model $\mathcal{M}$ then*

$$\mathcal{M} \models \alpha \quad \textit{iff} \quad \pi_{sem}(\mathcal{M}) \models \pi_{syn}(\alpha).$$

Proof The cases where α is of the form $t_1 = t_2$ or $t_1 \leq t_2$ are trivial. The non-trivial case is α of the form

$$\text{TRUE}(t_1, t_2, p(c_1, \ldots, c_n)).$$

We need only consider this case where all of the terms are ground, *i.e.*, variable free, since the formulae of STL and BTK are built up in an identical manner and the universes over which the quantified variables can range are identical. (α is a sentence so all variables are quantified.) $\pi_{syn}(\alpha)$ is of the form

$$p(t_1, t_2, \pi_{syn}^{[t_1,t_2]}(c_1), \ldots, \pi_{syn}^{[t_1,t_2]}(c_n)).$$

$\pi_{sem}(\mathcal{M})$ will be a model for this sentence iff

$$\langle t_1^\sigma, t_2^\sigma, \pi_{syn}^{[t_1,t_2]}(c_1)^\sigma, \ldots, \pi_{syn}^{[t_1,t_2]}(c_n)^\sigma\rangle \in p^\sigma,$$

where σ is the interpretation function of $\pi_{sem}(\mathcal{M})$. By definition, $\mathcal{M}$ is a model of α iff

$$\langle M[M(t_1), M(t_2), c_1], \ldots, M[M(t_1), M(t_2), c_n]\rangle \in M_5[M(t_1), M(t_2), p].$$

Clearly from the construction of $\pi_{sem}(\mathcal{M})$ all temporal terms are given the same denotation in BTK as in STL, *i.e.*, $t^\sigma = M(t)$ for all temporal terms t. We also claim that all non-temporal terms, *in a given TRUE context*, are given the same denotation in BTK as in STL. If the term is a constant this follows directly from the definition of π_{sem}, *i.e.*, $c^\sigma = M_2(c)$. If the term is of the form $f(c_1, \ldots, c_n)$ and is within the temporal context determined by temporal terms t_1 and t_2, then if we take $(c_i)^\sigma = M(M(t_1), M(t_2), c_i)$ for all i by induction, then

$$\begin{aligned}
&\pi_{syn}[f(c_1, \ldots, c_n)]^\sigma \\
&= [f(t_1, t_2, \pi_{syn}^{[t_1,t_2]}(c_1), \ldots, \pi_{syn}^{[t_1,t_2]}(c_n))]^\sigma \\
&= f^\sigma((t_1)^\sigma, (t_2)^\sigma, \pi_{syn}^{[t_1,t_2]}(c_1)^\sigma, \ldots, \pi_{syn}^{[t_1,t_2]}(c_n)^\sigma) \\
&= [M_4(M(t_1), M(t_2), f)][M(M(t_1), M(t_2), c_1), \ldots, \\
&\quad M(M(t_1), M(t_2), c_n)] \\
&= M(f(c_1, \ldots, c_n)).
\end{aligned}$$

Hence all of the terms are given an identical denotation. But using the definition of p^σ we have that

$$\langle t_1^\sigma, t_2^\sigma, \pi_{syn}^{[t_1,t_2]}(c_1)^\sigma, \ldots, \pi_{syn}^{[t_1,t_2]}(c_n)^\sigma \rangle \in p^\sigma \quad \text{iff}$$
$$\langle M(t_1), M(t_2), M[M(t_1), M(t_2), c_1], \ldots,$$
$$M[M(t_1), M(t_2), c_n] \rangle \in p^\sigma \quad \text{iff}$$
$$\langle M[M(t_1), M(t_2), c_1], \ldots, M[M(t_1), (t_2), c_n] \rangle$$
$$\in M_5[M(t_1), M(t_2), p]$$

Q.E.D. ■

Corollary 2 *A sound proof theory in BTK can be used to produce sound inferences in STL.*

Proof Let α and β be sentences of STL. We claim that if $\pi_{syn}(\alpha) \vdash \pi_{syn}(\beta)$ is a sound deduction in BTK, then $\alpha \models \beta$ in STL. That is, if the syntactic transformation of α can be used to deduce (soundly) the syntactic transformation of β then α entails β in STL.[13]

If $\pi_{syn}(\alpha) \vdash \pi_{syn}(\beta)$ then, by the assumption of soundness, for all BTK models, $\mathcal{M}'$, we have that $\mathcal{M}' \models \pi_{syn}(\alpha)$ implies that $\mathcal{M}' \models \pi_{syn}(\beta)$. Thus, this also holds for all models which have the special form $\pi_{sem}(\mathcal{M}_S)$ for all STL models $\mathcal{M}_S$. Hence, by theorem 1 we have that for all STL models $\mathcal{M}_S$, $\mathcal{M}_S \models \alpha$ implies $\mathcal{M}_S \models \beta$. In other words $\alpha \models \beta$ in STL. ■

References

[Allen, 1983] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[Allen, 1984] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[Barwise, 1977] Jon Barwise. *Handbook of Mathematical Logic.* North-Holland, Amsterdam, 1977.

[Dowty *et al.*, 1981] David R. Dowty, Robert E. Wall, and Stanley Peters. *Introduction to Montague Semantics.* Synthese Language Library. D. Reidel, Holland, 1981.

[Green, 1969] Cordell C. Green. Application of theorem proving to problem solving. In *Proceedings of the 1st IJCAI*, pages 219–239, 1969.

[Halpern and Shoham, 1986] J.Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. In *Proceedings of the Symposium on Logic in Computer Science*, 1986.

[Hayes and Allen, 1987]
Patrick J. Hayes and James F. Allen. Short time periods. In *Proceedings of the 10th IJCAI*, Milan, Italy, 1987.

[13] $\alpha \models \beta$ if any model which satisfies α (*i.e.*, assigns truth to α) also satisfies β. A proof theory is said to be sound if $\alpha \vdash \beta$ implies $\alpha \models \beta$. It is said to be complete if $\alpha \models \beta$ implies $\alpha \vdash \beta$.

[Herbrand, 1930] Jacques Herbrand. *Recherches sur la theorie de la demonstration.* PhD thesis, Dissertation, Paris, 1930.

[Ladkin, 1987] Peter B. Ladkin. The completeness of a natural system for reasoning with time intervals. In *Proceedings of the 10th IJCAI*, pages 462–467, 1987.

[Ladkin, 1988] Peter B. Ladkin. Satisfying first-order constraints about time intervals. In *Proceedings of the 7th AAAI*, pages 512–517, 1988.

[Lifschitz, 1987] Vladimir Lifschitz. A theory of action. In *Proceedings of the 10th IJCAI*, pages 966–972, 1987.

[McCarthy and Hayes, 1969] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*. Edinburgh University Press, 1969.

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

[Walther, 1987] Christoph Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation.* Research Notes in Artificial Intelligence. Pitman, London, 1987.

[Wang, 1952] Hao Wang. Logic of Many-Sorted Theories. *Journal of Symbolic Logic*, 17, 1952.

[Wos, 1988] Larry Wos. *Automated Reasoning: 33 Basic Research Problems.* Prentice-Hall, New Jersey, 1988.

A Simple Solution to the Yale Shooting Problem

Andrew B. Baker
Department of Computer Science
Stanford University
Stanford, CA 94305

Abstract

Most of the solutions proposed to the Yale shooting problem have either introduced new nonmonotonic reasoning methods (generally involving temporal priorities) or completely reformulated the domain axioms to represent causality explicitly. This paper presents a new solution based on the idea that since the abnormality predicate takes a situational argument, it is important for the meanings of the situations to be held constant across the various models being compared. This is accomplished by a simple change in circumscription policy: when *Ab* is circumscribed, *Result* (rather than *Holds*) is allowed to vary. In addition, we need an axiom ensuring that every consistent situation is included in the domain of discourse. Ordinary circumscription will then produce the intuitively correct answer. Beyond its conceptual simplicity, the solution proposed here has additional advantages over the previous approaches. Unlike the approach that uses temporal priorities, it can support reasoning backward in time as well as forward. And unlike the causal approach, it can handle ramifications in a natural manner.

1 Introduction

The formalization of reasoning about change has proven to be a surprisingly difficult problem. Standard logics are inadequate for this task because of difficulties such as the frame problem [McCarthy and Hayes, 1969]; nonmonotonic reasoning seems to be necessary. Unfortunately, as demonstrated by the Yale shooting problem of Hanks and McDermott [1987], the straightforward use of standard nonmonotonic logics (such as circumscription) for reasoning about action leads to counter-intuitive results.

There have been a large number of solutions proposed to the shooting problem [Gelfond, 1988, Haugh, 1987, Kautz, 1986, Lifschitz, 1987a, Lifschitz, 1987b, Morris, 1988, Pearl, 1988, Shoham, 1988, and others], but none of them are completely satisfactory. Some of these solutions cannot handle examples that require reasoning backward in time. And others require that the domain axioms be written in a rather restrictive format. This paper presents a new approach to the shooting problem that avoids these difficulties.

In the next section, we describe the shooting problem. Section 3 surveys some of the previous solutions and their limitations. Section 4 presents our solution, albeit in a slightly simplified form; this solution is refined in Section 5. In Section 6, we consider some additional temporal reasoning scenarios in order to compare the various approaches to the Yale shooting problem. Concluding remarks are contained in Section 7.

2 The shooting problem

The Yale shooting problem arises regardless of which temporal formalism is used; we will use the situation calculus [McCarthy and Hayes, 1969]. A *situation* is the state of the world at a particular time. Given an *action* a and a situation s, $Result(a, s)$ denotes the new situation after action a is performed in situation s. A truth-valued *fluent* (the only kind of fluent that will concern us) is a property that may or may not hold in a given situation. If p is a fluent and s is a situation, then $Holds(p, s)$ means that the fluent p is true in situation s.

With these conventions, one might use standard first-order logic to formalize the effects of various actions, but there are some well-known problems with this monotonic approach. The *frame problem* [McCarthy and Hayes, 1969], to which this paper will be limited, is that we would need to write down a great many axioms specifying those properties that are *un*changed by each action. And yet, intuitively, all of these frame axioms seem redundant; we would like to specify just the positive effects of an action, and then somehow say that nothing else changes. Part of the motivation behind the development of nonmonotonic reasoning was to formalize this notion, and thus to solve the frame problem; we will use *circumscription* [McCarthy, 1980, McCarthy, 1986]. If $\mathcal{A}$ is a formula, P is a predicate, and Z is a tuple of predicates and

functions, then the circumscription of P in $\mathcal{A}$ with Z varied is written as Circum$(\mathcal{A}, P, Z)$ [Lifschitz, 1985]. This formula selects those models of $\mathcal{A}$ in which the extension of the predicate P is minimal (in the set inclusion sense). Besides P, only those predicates and functions in Z are allowed to vary during this minimization process.

Consider the standard default frame axiom:[1]

$$\neg Ab(p, a, s) \Rightarrow (Holds(p, Result(a, s)) \equiv Holds(p, s)). \quad (1)$$

This says that the value of a fluent persists from one situation to the next unless something is abnormal. The original intention was to circumscribe Ab with $Holds$ varied. (We will refer to this as the *standard* circumscription policy.) It was hoped that this minimization of abnormality would ensure that a fluent would persist unless a specific axiom forced this fluent to change.

Unfortunately, this approach does not work. In a sequence of events, often one can eliminate an expected abnormality at one time by introducing a totally gratuitous abnormality at another time. In this case, there will be multiple minimal models only one of which will correspond to our intuitions.

The standard example, of course, is the Yale shooting problem [Hanks and McDermott, 1987].[2] In this problem, which we are simplifying slightly from the Hanks and McDermott version, there are two fluents, *Loaded* and *Alive*, and two actions, *Wait* and *Shoot*. The story is that if the gun is shot while it is loaded, a person (named Fred) dies. There are no axioms about *Wait*, so the general-purpose frame axiom should ensure that it does not change anything. In the original situation, the gun is loaded, and Fred is alive. If the actions *Wait* and then *Shoot* are performed in succession, what happens? Let $\mathcal{YSP}$ be the conjunction of the default frame axiom (1) with the following domain axioms:

$$Holds(Loaded, s) \Rightarrow \neg Holds(Alive, Result(Shoot, s)), \quad (2)$$

$$Holds(Loaded, S0), \quad (3)$$

$$Holds(Alive, S0). \quad (4)$$

What does

$$\text{Circum}(\mathcal{YSP}; Ab; Holds)$$

have to say about the truth value of

$$Holds(Alive, Result(Shoot, Result(Wait, S0)))?$$

We might guess that the waiting has no effect, and thus the shooting kills Fred, but circumscription is not so cooperative. Another possibility according to circumscription is that the gun mysteriously becomes unloaded while waiting, and Fred survives. This second model contains an abnormality during the waiting that was not present in the first model, but there is no longer an abnormality during the shooting. (Since the gun is unloaded, Fred does not change from *Alive* to not *Alive*.) So both models are minimal, and the formalization must be altered in some way to rule out the anomalous model.[3]

3 Previous approaches

There have been a large number of solutions proposed to the shooting problem. This section discusses the two most popular groups of solutions.

3.1 Chronological minimization

One idea, proposed in various forms by Kautz [1986], Lifschitz [1987b], and Shoham [1988], is *chronological minimization* (the term is due to Shoham). This proposal claims that we should reason forward in time; that is, apply the default assumptions in temporal order. So in the shooting scenario, we should first conclude that the waiting action is not abnormal. Then, since the gun would remain loaded, we would conclude that Fred dies. Each of the above authors successfully constructs a nonmonotonic logic that captures this notion of chronological minimality. Kautz, for example, uses a modified version of circumscription in which abnormalities at earlier times are minimized at a higher priority than those at later times.

While this approach does in fact give the intuitively correct answer to the Yale shooting problem, it is nevertheless highly problematic. Its applicability seems to be limited to what Hanks and McDermott call *temporal projection* problems, or in other words, problems in which given the initial conditions, we are asked to predict what will be the case at a later time. One can also consider temporal *explanation* problems [Hanks and McDermott, 1987], i.e., problems requiring reasoning backward in time. For problems of this sort, chronological minimization generally does not work very well. (See the example in Section 6.1.) For this reason, chronological minimization is not a completely satisfactory solution.

3.2 Causal minimization

Another approach, developed by Haugh [1987] and by Lifschitz [1987a], is that of *causal minimization*. This method represents causality explicitly by stating that a fluent changes its value if and only if a successful

[1] Lower case letters represent variables. Unbound variables are implicitly universally quantified.

[2] A similar scenario, involving the qualification problem rather than the frame problem, was discovered independently by Lifschitz and reported by McCarthy [1986].

[3] The current formalization admits a third possibility: Fred might die during the waiting phase. Our solution will rule out this model also.

action causes it to do so. The intuition here is that there is a crucial difference between the abnormality of a gun becoming unloaded while waiting, and the abnormality of Fred dying when shot with a loaded gun: there is a *cause* for the second while the first is totally arbitrary. We will discuss the system of [Lifschitz, 1987a]. There, the effects of actions are represented with a predicate $Causes(a, p, v)$ that indicates that if the action a is successful, then the fluent p takes on the value v. (The success or failure of an action is determined by a *Precond* predicate that we will not discuss.) With this formalism, one specifies all the known causal rules (for the current example, $Causes(Shoot, Alive, False)$), and then circumscribes *Causes* with *Holds* varied. Since *Causes* does not take a situational argument, there obviously cannot be a conflict in minimizing it in different situations. Therefore, the shooting problem cannot arise.

The main drawback of this proposal is that it does not allow us to write our domain axioms in unrestricted situation calculus. Instead, we must use the *Causes* predicate. This is a severe restriction on our expressive power because there is simply no way to use the *Causes* predicate to express ramifications, domain constraints, or general context-dependent effects. (See Section 6.2 for a discussion of this issue.) In light of this difficulty, it would be useful to solve the Yale shooting problem in the original formalism.

4 The solution

Our approach to the Yale shooting problem consists of two innovations. First of all, when we circumscribe *Ab*, instead of letting the *Holds* predicate vary, we will let the *Result* function vary. That is, we will not think of $Result(Wait, S0)$ as being a fixed situation, with circumscription being used to determine which fluents hold in this situation. Instead, we will assume that for each combination of fluents, there is some possible situation in which these fluents hold. Circumscription will then be used to determine which of these situations might be the result of waiting in $S0$. Second, in order for this idea to succeed, we must already have every consistent situation in the domain of discourse; an axiom will be added to accomplish this.

To see why this approach makes sense, let us resist the temptation of appealing to causality or temporal priorities, and instead think about the problem in its own terms. Why is it that we prefer the model in which Fred dies to the one in which waiting unloads the gun? After all, if by making the world behave more abnormally in $S0$, we could make it behave less abnormally somewhere else, this would seem to be a fair trade. The problem is that if waiting unloads the gun, the resultant situation is a *different* situation than it would have been, so it is not the case that the world behaves more normally somewhere else. In our preferred model, the abnormality was that *Alive* changed to not *Alive* when *Shoot* was performed in a situation in which the gun was loaded. In the anomalous model, the world does not behave more normally in this situation; this situation just never comes about! But the usual circumscription policy, with *Holds* varied and *Result* fixed, completely misses this subtlety. It views the $Result(Wait, S0)$'s in the two models as the same situation, even though different fluents hold in them.

From this perspective, the shooting problem arises from the failure to index abnormality correctly; by varying *Result* instead of *Holds*, we can correct this problem. First, some details need to be discussed.

We will use a many-sorted language with object variables of the following sorts: for situations $(s, s_1, s_2, \ldots)$, for actions $(a, a_1, a_2, \ldots)$, for primitive fluents[4] $(p, p_1, p_2, \ldots)$, and for times $(t, t_1, t_2, \ldots)$. Times are integers, and the function $Time(s):t$ maps a situation to its time.[5] We have the primitive-fluent constants, *Alive* and *Loaded*, the situation constant, $S0$, and the action constants, *Wait* and *Shoot*. Finally, we have the predicate constant $Holds(p, s)$ and the function constant $Result(a, s):s$. Axioms (1)–(4) should be interpreted relative to these declarations.

Actions always increment the time:

$$Time(Result(a, s)) = Time(s) + 1. \tag{5}$$

We also need a uniqueness of names axiom:

$$Wait \neq Shoot. \tag{6}$$

Now, for the important part. Suppose that for every point in time, and for every possible combination of fluents, there is some situation at this time in which these fluents hold. We will discuss how to achieve this in general in the next section, but for the Yale shooting problem we add the following *existence of situations* axiom:

$$\begin{aligned}
&\exists s\ (Time(s) = t \wedge Holds(Alive, s) \\
&\quad \wedge Holds(Loaded, s)) \\
\wedge\ &\exists s\ (Time(s) = t \wedge Holds(Alive, s) \\
&\quad \wedge \neg Holds(Loaded, s)) \\
\wedge\ &\exists s\ (Time(s) = t \wedge \neg Holds(Alive, s) \\
&\quad \wedge Holds(Loaded, s)) \\
\wedge\ &\exists s\ (Time(s) = t \wedge \neg Holds(Alive, s) \\
&\quad \wedge \neg Holds(Loaded, s)).
\end{aligned} \tag{7}$$

[4] Section 5 will make use of *generalized* fluents; these are built up from the primitive fluents by using *And* and *Not*. Primitive fluents are simply those that do not contain logical connectives.

[5] We are using this syntax to keep explicit the special role of time; we could just as easily make time an ordinary fluent. Actually, the only reason time is used in this paper is to rule out "circular" models, in which a sequence of actions performed in one situation leads back to that same situation.

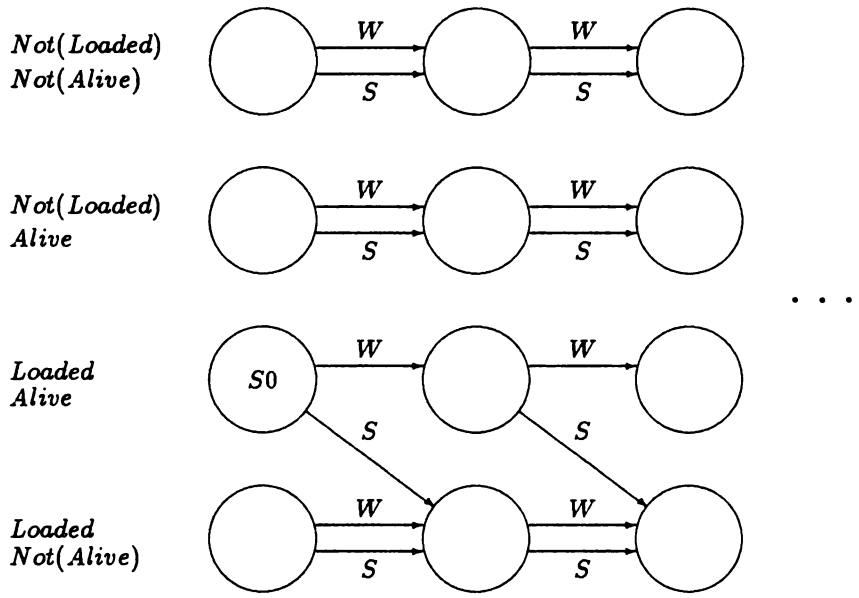


Figure 1: A minimal model of the Yale shooting problem

Let $\mathcal{A}$ be the conjunction of Axioms (1)–(7), and circumscribe Ab in $\mathcal{A}$ with $Result$ varied:

$$\mathcal{B} \equiv \text{Circum}(\mathcal{A}; Ab; Result).$$

Figures 1 and 2 are pictorial representations of models of $\mathcal{A}$. Time flows horizontally, and each circle represents the set of situations at a given time in which the fluents to its left either hold or do not hold as indicated. Axiom (7) ensures that each circle contains at least one situation. The W arrows show the result of performing the $Wait$ action, and the S arrows show the result of performing the $Shoot$ action. Diagonal arrows represent actions that change at least one fluent, and hence are associated with at least one abnormality.

Since the S arrows from situations in which $Loaded$ and $Alive$ hold lead to situations in which $Alive$ does not hold, these arrows must be diagonal. In the model of Figure 1, these are the *only* abnormalities, and therefore this is a minimal model of $\mathcal{A}$ and hence a model of $\mathcal{B}$. Figure 1 represents the expected model of the shooting problem. Figure 2, on the other hand, represents an unexpected model: one in which waiting in $S0$ unloads the gun. Note, however, that this added abnormality does not reduce the abnormality anywhere else. This model is strictly inferior to the previous one, and so it is not a model of $\mathcal{B}$. Therefore, our technique solves the Yale shooting problem.

Proposition 1 *$\mathcal{B}$ is consistent.*

Proof. A model of $\mathcal{B}$ corresponding to Figure 1 can be defined in a straightforward manner. □

Proposition 2 (Fred dies) $\mathcal{B} \models \neg Holds(Alive, Result(Shoot, Result(Wait, S0)))$

Proof. Consider a model of $\mathcal{A}$ where waiting in $S0$ unloads the gun; i.e., a model that satisfies

$$\neg Holds(Loaded, Result(Wait, S0)) \wedge Ab(Loaded, Wait, S0).$$

By (7), we can vary $Result$ to keep the gun loaded, and thus eliminate this abnormality without introducing any new abnormalities. (Axiom (6) ensures that $Result$ can be varied in this limited way without changing its effect for actions other than $Wait$.) Therefore, in all minimal models of $\mathcal{A}$ the gun remains loaded, and Fred dies. □

5 Existence of situations

In the last section, we violated the spirit of the nonmonotonic enterprise by adding an existence of situations axiom (7) that explicitly enumerated all possible situations. For more complicated problems, this axiom would be a bit unwieldy. In this section, we show how to write this axiom in a more general way.

We add a new sort to the language: generalized fluents (which are a supersort of primitive fluents), with the variables $f, f_1, f_2, \ldots$; and the functions $And(f, f):f$ and $Not(f):f$ to build up these fluents starting with primitive fluents. We will also alter the declaration $Holds(p, s)$ to $Holds(f, s)$ so the first argument can be any generalized fluent. We have:

$$Holds(And(f_1, f_2), s) \equiv Holds(f_1, s) \wedge Holds(f_2, s), \quad (8)$$

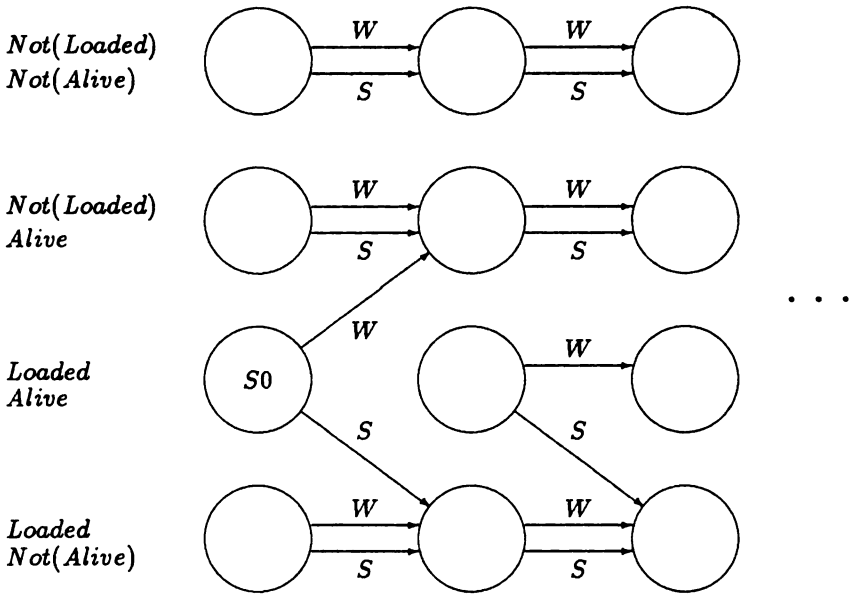


Figure 2: A nonminimal model of the Yale shooting problem

$$Holds(Not(f), s) \equiv \neg Holds(f, s), \quad (9)$$

$$And(f_1, f_2) \neq p, \quad (10)$$

$$Not(f) \neq p. \quad (11)$$

Axioms (8) and (9) are straightforward. Axioms (10) and (11) indicate that compound fluents do not belong to the subdomain of primitive fluents. Since the frame axiom only applies to primitive fluents, this rules out minimal models in which compound fluents persist at the expense of primitive ones.

For our existence of situations axiom, we would like to introduce a function $Sit(t, f) : s$ that maps a time and a fluent into some situation at that time in which the fluent holds:

$$Time(Sit(t, f)) = t \wedge Holds(f, Sit(t, f)).$$

This would guarantee that for each time, there is some situation such that $And(Alive, Loaded)$ holds, and one such that $And(Alive, Not(Loaded))$ holds and so on. But this would also mean that even inconsistent fluents like $And(Alive, Not(Alive))$ would be true in some situation; this contradicts (8) and (9). More generally, any domain constraint will render certain fluent combinations inconsistent. So instead, the existence of situations axiom will be written as a default rule:

$$\neg Absit(t, f) \Rightarrow (Time(Sit(t, f)) = t \wedge Holds(f, Sit(t, f))) \quad (12)$$

with *Absit* circumscribed.

In order for the circumscription of *Absit* to have its intended effect, some uniqueness of names axioms will be necessary. We will use an abbreviation from [Lifschitz, 1987a]: UNA$[f_1, \ldots, f_n]$, where $f_1, \ldots, f_n$ are (possibly 0-ary) functions, stands for the axioms:

$$f_i(x_1, \ldots, x_k) \neq f_j(y_1, \ldots, y_l)$$

for $i < j$ where f_i has arity k and f_j has arity l, and

$$f_i(x_1, \ldots, x_k) = f_i(y_1, \ldots, y_k) \Rightarrow (x_1 = y_1 \wedge \ldots \wedge x_k = y_k)$$

for f_i of arity $k > 0$. These axioms ensure that $f_1 \ldots, f_n$ are injections with disjoint ranges. We state that uniqueness of names applies to actions, fluents, and our special *Sit* function:

$$\text{UNA}[Wait, Shoot], \quad (13)$$

$$\text{UNA}[Alive, Loaded, And, Not], \quad (14)$$

$$\text{UNA}[Sit]. \quad (15)$$

(Axiom (13) is equivalent to Axiom (6).)

Let C be the conjunction of Axioms (1)–(5) and (8)–(15). In addition to circumscribing *Ab* as before, we now circumscribe *Absit* with *Holds*, *Time*, *Result*, and *Ab* allowed to vary:

$$\mathcal{D} \equiv \text{Circum}(C; Absit; Holds, Time, Result, Ab) \wedge \text{Circum}(C; Ab; Result).$$

Proposition 3 *$\mathcal{D}$ is consistent.*

Proposition 4 (Fred dies again) $\mathcal{D} \models \neg Holds(Alive, Result(Shoot, Result(Wait, S0)))$

The above approach for ensuring the existence of situations only works correctly for propositional fluents. If we had some fluent $Interesting(x)$, for instance, where x could range over integers, (12) would ensure the existence of a situation in which

$$And(Interesting(0), Interesting(1))$$

held, but with only finite conjunctions, it would not ensure the existence of a situation in which all integers were interesting. To do this, we would have to reify quantified formulas. Alternatively, we could use something like the following second-order existence of situations axiom:

$$\neg Absit2(t,h) \Rightarrow Time(Sit2(t,h)) = t \\ \wedge (Holds(p, Sit2(t,h)) \equiv h(p))$$

where h is a predicate variable.

6 Examples

In order to compare the different approaches to the Yale shooting problem, this section will discuss three additional temporal reasoning problems. Section 6.1 contains an example that requires reasoning backward in time. Section 6.2 discusses the issue of ramifications. And finally, Section 6.3 deals with changes that happen unexpectedly.

6.1 The murder mystery

Consider the following temporal explanation problem, which we will call the murder mystery. In this story, Fred is alive in the initial situation, and after the actions *Shoot* and then *Wait* are performed in succession (the opposite of the Yale shooting order), he is dead:

$$Holds(Loaded, s) \Rightarrow \\ \neg Holds(Alive, Result(Shoot, s)), \\ Holds(Alive, S0), \\ S2 = Result(Wait, Result(Shoot, S0)), \\ \neg Holds(Alive, S2).$$

The detective expert system is faced with the task of determining when Fred died, and whether or not the gun was originally loaded. If we used the obvious monotonic frame axioms, we would be able to conclude that the gun was originally loaded, and that Fred died during the shooting. Unfortunately, the standard circumscription policy is unable to reach this same conclusion. It has no preference for when Fred died, and even if it were told that Fred died during the shooting, it still would not conclude that the gun was originally loaded. Surely, assuming that the gun was loaded is the only way to explain the

$$Ab(Alive, Shoot, S0)$$

that would be entailed by Fred being shot to death, but circumscription is in the business of minimizing abnormalities — not explaining them.

Chronological minimization only makes the situation worse. It tries to delay abnormalities as long as possible, so it avoids any abnormality during the shooting phase, by postponing Fred's death to the waiting phase. It therefore concludes that the gun must have been *un*loaded![6]

Causal minimization yields the intuitive answer since it is only by assuming that the gun was originally loaded that it can explain the death without introducing an additional causal rule.

Our method also gives the right answer. The minimal model is the same one pictured in Figure 1, with $S2$ being reached by starting in $S0$ and following the S arrow and then the W arrow. There is a fine point, however. In addition to *Result*, the situation constants $S0$ and $S2$ also must be allowed to vary during the circumscription. In general with our approach, all situation constants and functions must be allowed to vary. This did not matter for the Yale shooting problem since in that problem, the situation $S0$ was fully specified.

6.2 Ramifications

Often, it is impractical to list explicitly all the consequences of an action. Rather, some of these consequences will be *ramifications*; that is, they will be implied by domain constraints [Ginsberg and Smith, 1988]. One of the main advantages of our method over causal minimization is that ours can handle ramifications, while causal minimization cannot.

A simple example of this limitation of causal minimization can be obtained from Hanks and McDermott's original version of the Yale shooting problem in which the gun was unloaded in the initial situation, and a *Load* action was performed before the waiting.[7] Using the notation from [Lifschitz, 1987a], we have the following causal specifications:

$$Causes(Load, Loaded, True), \\ Causes(Shoot, Alive, False).$$

Suppose that we added the fluent *Dead* and a domain constraint relating *Dead* and *Alive*:

$$Holds(Dead, s) \equiv \neg Holds(Alive, s). \qquad (16)$$

It would be nice if by minimizing *Causes* we could conclude that not only does shooting make Fred not alive; it also makes him dead:

$$Causes(Shoot, Dead, True). \qquad (17)$$

[6] Actually, even the standard circumscription policy says that the gun must be originally unloaded. Regardless of whether Fred dies during the shooting or the waiting, making the gun unloaded will keep Fred alive in other "possible" action sequences starting with $Result(Wait, S0)$. This, however, is just an artifact of the situation calculus.

[7] The example is from Matthew Ginsberg, personal communication.

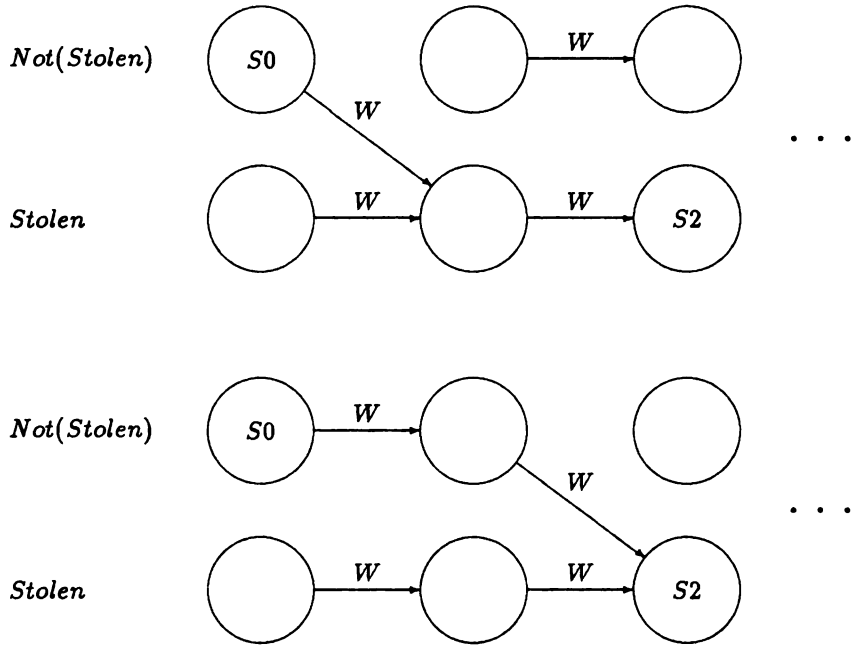


Figure 3: Two minimal models of the stolen car problem

Unfortunately, there is another causally minimal model in which *Load* kills Fred. In this model, there is no situation in which the gun is loaded while Fred is alive; therefore, (17) does not have to be added.

There have been some attempts at resolving this difficulty while remaining within the causal minimization framework, but so far none seem adequate. [Lifschitz, 1987a], for example, requires that fluents be divided into two groups, primitive and nonprimitive, with the causal laws and the frame axiom limited in application to the primitive fluents, and with the values of the nonprimitive fluents determined by their definitions in terms of the primitive fluents. So, in the above example, *Dead* would be a nonprimitive fluent.[8] The problem with this is that the fluents that change as the result of domain constraints need not be definitional in nature like *Dead*. Suppose two objects are connected in some manner. Moving either one of them will cause the other to move as well, so each object position will have to be primitive for some actions and nonprimitive for other actions. Furthermore, if the connection between the two objects is subsequently broken, then both positions will become primitive fluents. In short, the primitiveness of a fluent is dependent on both the situation and the action, and to correctly formalize this notion it seems that all the ramifications would have to be precomputed — a potentially intractable task [Ginsberg and Smith, 1988].

The approach advocated in this paper handles ramifications correctly. Domain constraints determine which situations can exist in the model; in the above example, there are only four types of situations that can exist at any time point: *Loaded* can be true or false, *Alive* can be true or false, and *Dead* must have the opposite value of *Alive*. The causal laws, like (2) further constrain the resultant situation of an action. So if the gun is loaded, and Fred is alive, then (2) will demand that Fred must be not alive after being shot, and the domain constraint (16) will ensure that Fred is dead in this resulting situation. Finally, the minimization of abnormality will keep the gun loaded since we have not axiomatized the notion of running out of bullets. The strange behavior of the causal approach cannot occur here, because we require all possible situations to exist in the model.

6.3 The stolen car problem

We consider one final example, Kautz's stolen car problem [Kautz, 1986]. In the initial situation, your car is not stolen. After you wait two times, it is stolen:

$$\neg Holds(Stolen, S0),$$
$$S2 = Result(Wait, Result(Wait, S0)),$$
$$Holds(Stolen, S2).$$

[8] In this paper, we also use primitive and nonprimitive fluents, but for a somewhat different purpose. As discussed in the next paragraph, we have no difficulty in letting both *Alive* and *Dead* be primitive fluents.

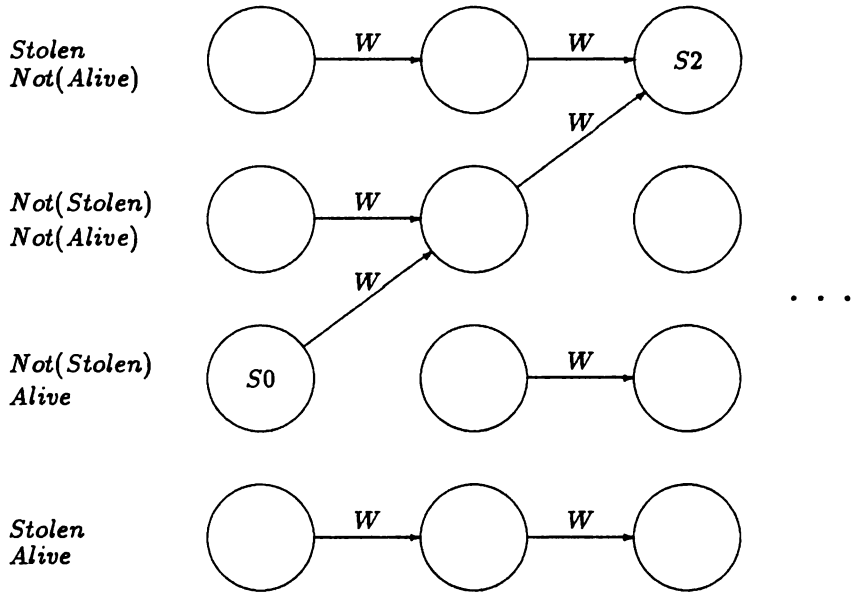


Figure 4: A peculiar minimal model of the modified stolen car problem

Was the car stolen during the first waiting phase or the second?

This problem differs in character from those discussed previously: if we formalized this using the monotonic frame axioms, we would have a contradictory theory. Nevertheless, the problem seems simple enough; the car could have been stolen during either of the two waiting phases, and there is no reason to prefer one over the other. And indeed, the standard circumscription policy will yield models corresponding to both of these alternatives. But as pointed out in [Kautz, 1986], chronological minimization will claim that the car must have been stolen during the second waiting action. This is clearly unreasonable.

The basic causality-based formalism cannot deal correctly with this problem either. Since it only allows changes that are caused, it must assume that waiting *causes* the car to be stolen. So not only is the car stolen in the first waiting phase, it will always be stolen whenever you wait. Several authors have augmented the causal formalism to better address problems of this sort. Lifschitz and Rabinov [1988], for example, allow "miracles," i.e., changes that cannot be explained. When their new predicate *Miracle* is minimized at a lower priority than *Causes*, they are able to conclude that a miracle occurred during either the first waiting action or during the second. Another approach, proposed by Morgenstern and Stein [1988], explains unexpected changes by assuming that additional actions (selected from a list of known action types) have been performed, possibly in parallel with the actions that we know about. So in this case, if they had an appropriate causal rule for the action *Steal*, Morgenstern and Stein would be able to conclude that a *Steal* action had been performed during one of the waiting phases.

Finally, the solution proposed in this paper also runs into some difficulties with the stolen car problem. It does correctly handle the simple version. Figure 3 shows that, as desired, both possibilities (the car being stolen during either the first or the second waiting phase) are minimal models. Unfortunately, if we have another fluent, say *Alive*, such that

$$Holds(Alive, S0),$$

besides the two expected minimal models, there will be an additional one (shown in Figure 4) that satisfies:

$$Ab(Alive, Wait, S0)$$
$$\wedge\, Ab(Stolen, Wait, Result(Wait, S0)).$$

Not only is the car stolen during the second waiting action, poor Fred dies during the first waiting action![9] In this model, waiting steals the car in a situation in which Fred is dead; this abnormality is different from the car being stolen while Fred is still alive.

Some words of explanation are in order. First of all, this third model may be warranted in some cases. If Fred were the security guard, then it is plausible that the thief killed him before stealing the car. Of course, for two arbitrary fluents, it is a bit silly to posit an

[9] This was pointed out by Matthew Ginsberg.

abnormality of one fluent — not to get rid of an abnormality of the second fluent — but merely to change this abnormality into another abnormality that is "almost the same." But since circumscription's definition of a minimal model is based on set inclusion, circumscription does not recognize the concept of "almost the same."

Secondly, the Ab facts may be thought of as highly specific causal rules. Thus, for example,

$$Ab(Alive, Wait, S0)$$

might be interpreted as saying that $Wait$ causes the value of $Alive$ to change if it is performed in a particular situation — one that is at time 1 in which Fred is alive and the car is not stolen. So in order to explain unexpected changes, our approach, rather than assuming the occurrence of additional actions (as in [Morgenstern and Stein, 1988]) or miracles (as in [Lifschitz and Rabinov, 1988]), instead extends the causal theory in a minimal (and admittedly peculiar) fashion. This is obviously *not* the right thing to do. There is no reason, however, why the improvements to causal minimization cannot also be integrated with the current approach.

7 Conclusion

This paper has presented a new approach to nonmonotonic temporal reasoning. The approach correctly formalizes problems requiring reasoning both forward and backward in time, and it allows for some of an action's effects to be specified indirectly using domain constraints.

It should be noted that, in a certain sense, our solution works for the same reason that causal minimization does. Causal minimization is not tempted to unload the gun because it is minimizing the extent of the *Causes* predicate rather than actual changes in the world; unloading the gun would prevent the $Causes(Shoot, Alive, False)$ fact from *being used*, but it would not eliminate the fact itself. Similarly, our solution minimizes even those abnormality facts associated with situations that do not really happen. But since we stick with the standard axioms (rather than introducing a special *Causes* predicate), our approach appears to be the more robust of the two: for us, even those abnormalities that arise as ramifications will not be eliminated by the assumption of gratuitous changes.

There are several possible directions for future work. One project would be to extend this approach to handle more expressive models of time, of action, and of causality. Another question is how to augment the formalism with default rules concerning the typical values of fluents; we might, for instance, wish to assume that guns are loaded by default. But if we add defaults of this kind carelessly, the Yale shooting problem is likely to reappear. Lastly and most importantly, there is the issue of implementation. It would be interesting if there were a formulation of this paper's theory that, when given to an "off the shelf" circumscriptive theorem prover, would lead to chains of reasoning that were both intuitive and efficient. There has been some preliminary work on these topics, but much more remains to be done.

Acknowledgements

I would like to acknowledge helpful discussions with Matthew Ginsberg, Vladimir Lifschitz, Karen Myers, Arkady Rabinov, Yoav Shoham, and David Smith. I especially thank Matthew Ginsberg for his comments on several drafts of this paper and for his encouragement.

The author is supported by an AFOSR graduate fellowship.

References

[Gelfond, 1988] Michael Gelfond. Autoepistemic logic and formalization of common-sense reasoning. In *Proceedings of the Second International Workshop on Non-Monotonic Reasoning*, Munich, 1988.

[Ginsberg and Smith, 1988] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35:165–195, 1988.

[Hanks and McDermott, 1987] Steve Hanks and Drew McDermott. Nonmonotonic logics and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[Haugh, 1987] Brian A. Haugh. Simple causal minimizations for temporal persistence and projection. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 218–223, 1987.

[Kautz, 1986] Henry A. Kautz. The logic of persistence. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 401–405, 1986.

[Lifschitz and Rabinov, 1988] Vladimir Lifschitz and Arkady Rabinov. Miracles in formal theories of action. Draft, 1988. To appear in *Artificial Intelligence*.

[Lifschitz, 1985] Vladimir Lifschitz. Computing circumscription. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 121–127, 1985.

[Lifschitz, 1987a] Vladimir Lifschitz. Formal theories of action. In Matthew L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*. Morgan Kaufmann, Los Altos, CA, 1987.

[Lifschitz, 1987b] Vladimir Lifschitz. Pointwise circumscription. In Matthew L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*. Morgan Kaufmann, Los Altos, CA, 1987.

[McCarthy and Hayes, 1969] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[McCarthy, 1980] John McCarthy. Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[McCarthy, 1986] John McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[Morgenstern and Stein, 1988] Leora Morgenstern and Lynn Andrea Stein. Why things go wrong: A formal theory of causal reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 518–523, 1988.

[Morris, 1988] Paul H. Morris. The anomalous extension problem in default reasoning. *Artificial Intelligence*, 35:383–399, 1988.

[Pearl, 1988] Judea Pearl. On logic and probability. *Computational Intelligence*, 4:99–103, 1988.

[Shoham, 1988] Yoav Shoham. Chronological ignorance: Experiments in nonmonotonic temporal reasoning. *Artificial Intelligence*, 36:279–331, 1988.

Belief, Metaphorically Speaking

John A. Barnden

Computing Research Laboratory
New Mexico State University
Las Cruces, NM 88003

Abstract

The central claim of the paper concerns AI systems that attempt to represent propositional attitudes in realistic situations, and particularly in situations portrayed in natural language discourse. The claim is that the system, in order to achieve a coherent, useful view of a situation, must often ascribe, to outer agents, views of inner agents' attitudes that are based on rich explications in terms of commonsense metaphorical views of mind. This elevates the emasculated metaphors based on notions of world, situation, container, and so on that underlie propositional attitude representation proposals to the status of explicitly-used, rich metaphors. A system can adopt different patterns of commonsense inference about attitudes by choosing different metaphors. The current stage of development of a detailed representation scheme based on the claim is described. The scheme allows different metaphors to be used for the explication of attitudes at different levels in a nested-attitude situation.

1 Introduction

I argue for a major shift in the way in which beliefs and other propositional attitudes are represented and reasoned about. The shift is not predominantly at the level of the syntactic nature of the logic or other representational framework used. Rather, it is at the higher level of the *content* of the representational structures, and involves "explicating" attitudes in accordance with *the commonsense models or theories that people actually seem to use for thinking and talking about attitude states.* Moreover, commonsense theories of propositional attitudes are largely metaphorical, so that the shift is towards a tight integration of the study of attitudes with the study of *metaphor* (as a cognitive phenomenon rather than a superficially linguistic one).

The research reported here begins to grapple seriously with propositional attitudes as they arise in realistic natural language communication and in realistic commonsense reasoning, and integrates the study of propositional attitudes more closely with various other areas, notably the following: recent work in linguistics on commonsense views of propositional attitudes and other abstract matters [e.g. Johnson, 1987, Lakoff, 1987, Sweetser, 1987, Sweetser, forthcoming, Talmy 1988]; plausible, commonsense reasoning more generally; and discourse coherence theory. In particular, the research is related to the work of Fass [1987], Hobbs [1983a,b, 1985] and Carbonell [1982] on discourse coherence, metaphor and their relationship. See also [Johnson, 1987: pp.5ff, 37]. In this context, my explication of propositional attitude states in commonsense/metaphorical terms is a special case of what Hobbs [1985] calls "elaboration".

The shift of approach put forward by this paper concerns an aspect of nested attitudes that is not commonly addressed. Consider the sentence, "Bob hopes that Mike realizes that Jim's wife is clever". The typical theoretical concern with this sentence would be the variety of ways in which the phrase "Jim's wife" could contribute to the meaning of the whole. I focus, on the other hand, on Mike's *state of realization* itself. I claim that we need to pay serious attention to the question of what commonsense view of that state Bob is likely to be entertaining. I will argue below that, in an example like the one just given, Mike's hoped-for state of realization is quite likely to need explication (elaboration, decomposion) in the terms of some commonsense and probably metaphorical view of mind that the surrounding discourse suggests is being used by Bob in his view of Mike. This need arises from a need to establish coherence between the interpretation of the sentence and that of surrounding sentences.

It would be wrong to conclude, however, that the research concerns propositional attitudes only in so far as they are reported in natural language utterances. On the contrary, I claim that people think about attitudes in terms of a variety of commonsense and largely metaphorical models, and that the models that are implicit or explicit in discourse are a reflection of them. Thus, the approach concerns propositional attitude representation in general. However, it is convenient in an exposition to discuss examples in the context of natural language processing.

It is possible that there are non-metaphorical, yet commonsensical, ways of explicating mental states. One

such may be an explication of a particular belief in terms of a disposition to act in a certain way. My approach is intended to encompass such possibilities, but for brevity I talk below as if commonsensical explications are all metaphorical, as I do think the metaphorical ones are in the majority.

2 Metaphors of Mind

It is typical for people to think commonsensically of minds by means of metaphors, many of which are fairly standard. Such prevalent, commonsense, metaphorical views of mind have received some close attention [e.g.: Johnson, 1987, Lakoff & Johnson, 1980, Katz *et al.*, 1988, Larsen, 1987, Reddy, 1979, Sweetser, 1987, Sweetser, forthcoming, Tomlinson 1986]. Some metaphors for mind are closely related to (if not identical to) common metaphors for arguments and reasoning processes — consider for instance the ARGUMENT-AS-WAR, ARGUMENT-AS-BUILDING, and ARGUMENT-AS-JOURNEY metaphors analysed in detail in Johnson [1987] and Lakoff & Johnson [1980]. Metaphorical relationships between understanding and seeing have been studied [e.g. Johnson, 1987: p.108, Sweetser, forthcoming]. The metaphors used are also linked to common(sense) metaphors for language, particularly the famous CONDUIT metaphor for communication [Reddy, 1979] and bodily-force-based metaphors for moral action, alethic modalities (necessity, possibility), and reasoning processes [Johnson, 1987: p.16, 48ff, Sweetser, forthcoming, Talmy, 1988]. A consideration of perfectly mundane discourse readily reveals that people think of minds as BATTLEGROUNDS, CONTAINERS, MINES (as in gold-mine), SOCIETIES, LANDSCAPES, WORLDS, MACHINES, COMPUTERS, and many other things. Consider for instance the following fragments, with annotations about the metaphors used:

> She has it firmly fixed in her mind that John is evil. [MIND-AS-CONTAINER]
>
> It gradually became clear to him that she loved him. His vision had been obscured by resentment. [UNDERSTANDING-AS-SEEING]
>
> Her wishful thinking battled with the frightening truth for some time. [MIND-AS-BATTLEGROUND]

These sentences rely on systematic metaphors that are appealed to by utterer and understander alike. I take the MIND-AS-BATTLEGROUND metaphor as my prime example, even though Sweetser [1987] gives primacy to other metaphors. None of my claims rest on any special quality of that metaphor.

We can even claim that people think of minds *only* in such metaphorical terms — that they do not have access to an objective, metaphor-independent, commonsense theory of mind for which the metaphors merely supply a colorful way of talking or thinking. In short, the metaphors are *constitutive* of people's views of mind. (See, e.g., [Black, 1954/5, Johnson, 1987: incl. p.xii,5,112, Richards, 1936] for constitutive metaphors in general.) A fortiori, the metaphors are *creative*, in the sense of supplying (possibly misleading) information that would not otherwise be available. The creative aspect goes beyond the mere establishment of an analogy between some structure in the metaphor vehicle and some *independently*-known structure in the metaphor tenor [e.g. Black, 1954/5, Hobbs, 1983a,b, Johnson, 1987: p.69].

Actually, I rely only on a slightly weaker claim than constitutivity, namely that people's patterns of inference, about minds or whatever else, are *frequently* governed by metaphors [e.g.: Carbonell, 1982, Johnson, 1987: esp. circa p.112, Lakoff, 1987, Lakoff & Johnson, 1980]. My approach nevertheless allows for the possibility that people sometimes think about minds in a metaphor-free way.

3 The Benefits of Metaphorical Explication

3.1 Discourse Coherence

Consider the following discourse fragment:

(1) *Professor Jenny Zorn hopes that Xavier will come to realize that his theory is faulty.*

(2) *But she's afraid that the idea of its being faulty is battling against his habit of wishful thinking.*

How could an AI natural language understanding system give this fragment a coherent interpretation? Before answering this question, we should attend to a scope ambiguity in the second sentence. That sentence could be taken to mean that Jenny Zorn is herself thinking in terms of of the metaphor of an idea battling with a habit (the inner-scope reading of the metaphor). Alternatively, we could assume merely that the speaker is using the metaphor to describe Xavier's hoped-for mental state (the outer-scope reading). Without wishing to deny the possible importance of the latter reading, I hold that the former is very likely in practice to be the more appropriate one, and that therefore we must be seriously concerned with treating it properly.

I claim, then, that the system can best give a *coherent* interpretation to sentence (1) and sentence (2) (assuming inner scope) by giving the "coming to realize" notion in the first sentence a *metaphorical explication* (elaboration, expansion) in terms of the MIND-AS-BATTLEGROUND metaphor appealed to in the second sentence. That is, the system should give sentence (1) an interpretation similar to what it would have given, say, the following sentence in the same context:

(3) *Zorn hopes that the idea of Xavier's theory being faulty will gain dominance in the battleground of his mind.*

This is the most straightforward way for the system to respect the plausible, inner-scope interpretation of (2) that Zorn is actually thinking about Xavier by means of the MIND-AS-BATTLEGROUND metaphor. And it is the most straightforward way for the system to be in a position to draw the full plausible implications of the discourse.

One such implication is that Zorn thinks, despite her hope, that there is a very good chance that Xavier will *not* come to realize that his theory is faulty, and, further, that the reason for that possible failure is Xavier's wishful thinking. The connotation of "X is battling against Y", taken in a literal sense, is that it is quite likely, or at least very possible, that X will lose. This connotation, i.e. plausible conclusion, carries over to metaphorical battles as well. In the present example, the possibility of X losing (i.e. the idea of the theory being faulty losing) is reinforced by the phrase "But she's afraid that", which could itself be taken suggesting the possibility that there is something working against Zorn's hope. Of course, there are alternative possibilities suggested by that phrase. Consider that the second sentence could have been

> *But she's afraid that if he does he might have a nervous breakdown.*

In this case the contrast underlying the "But" is that the hoped-for event may have an undesirable consequence, rather than that the hope may be unrealistic, as in our actual example. This observation shows that the phrase "But she's afraid that" is not enough by itself to lead the system to high confidence in the hypothesis that Zorn thinks that Xavier may well not come to realize his theory is faulty. It is the content of the rest of sentence (2) that firmly supports that hypothesis.

But how does the system use that content to support the hypothesis? It can only do so if it appeals to a metaphorical correspondence under which the process of the idea X winning (or losing) the battle corresponds to the process of Xavier coming (respectively, failing to come) to realize X. Furthermore, a perception of the correspondence should be attributed to Zorn, since Zorn is meant to be thinking about Xavier in the terms of the metaphor, and it is her views and inferences that are the subject of the system's inferences. In sum: what Zorn hopes for is viewed by Zorn herself as being metaphorically expressible (in a mental sense) as the idea X winning the battle. But this is just to say that we can conclude (at least very plausibly) that Zorn hopes that, in metaphorical terms, the idea will win the battle, as hypothetical sentence (3) says. (Note that in none of this do we assume that Zorn fails to realize that she is indulging in metaphorical thought.) One might claim that it is in principle possible for Zorn to have the hope expressed by (1) and to be thinking about Xavier in terms of the metaphor *without* making the necessary connection to the notion of the idea X winning the battle; we can grant this but still hold that normally the system should presume that she does make the connection.

This argument leaves open the possibility that the system should not only maintain an internal expression E_3 paraphrasable as (3), but should *also* maintain an internal expression E_1 that expresses (1) without appeal to any metaphor; indeed, this seems a very natural suggestion, since E_1 would be what the system would produce anyway on encountering (1) and before encountering (2). In that case, E_3 would be best viewed as merely a plausible inference from E_1, rather than as actually being, as I claim, the rendering of (1) that the system should eventually choose. Further, one might raise the following larger question at this point:

> "Even granting that context sometimes mandates the production of expressions like E_3 as an inference from sentences like (1), surely in most cases it does not (i.e. no metaphor is suggested): so why not use, in *most* cases, existing representational tools, which do not make metaphorical ascriptions? Why bother to develop a representational scheme based on the metaphorical-explication claim, since it pertains only to an unusual effect the study of which can be postponed?"

My response both to this question and to the idea that even in the presence of E_3 there should be a separate, non-metaphorical expression E_1 is mainly as follows.

In cases where no metaphor is (so far) suggested by context, the proposed representational scheme's rendering of a sentence like (1) is based on giving both the inner and outer attitude an explication in terms of a certain default metaphor. The explication is such that the representational expressions yielded are much like those used in certain *existing* approaches. The default metaphor is a simple "AGENT'S-WORLD" metaphor, which can be seen as a sort of MENTAL-SPACE metaphor, in the Fauconnier [1985] sense of mental space. As a result, the default representational constructs are akin to, and not appreciably more complex than, those used in existing approaches based on partitions, environments, or mental spaces [see particularly: Dinsmore, 1988, Fauconnier, 1985, Wilks & Ballim, 1987, Wilks & Bien, 1983]. This can be seen from the expressions to be presented in Section 5. So, on encountering (1) the system produces an interpretation based on the default metaphor, instead of a non-metaphorical interpretation E_1. In a case where a non-default metaphor is suggested, as by sentence (2), the system suppresses any default-metaphor interpretation, and instead adopts one, like E_3, based on the suggested metaphor. Although it might sometimes be possible to keep the default-metaphor interpretation active, as well as the non-default one, this is not generally desirable because different mental metaphors can lead to different patterns of infer-

ence, as argued below.

Therefore, the system takes an approach that is based throughout on metaphorical views of mental states. The point is that not only does this remain faithful to the way people actually view mental states, but there is nothing lost by doing it since the representations based on the default metaphor are no more elaborate than the expressions in existing non-metaphorical schemes founded on notions of space, environment or partition. The approach does add something not present in those schemes, though: an explicit commitment to the idea that the represented agents themselves have views that are based on the world metaphor.

This response to the question is, I hold, sufficient by itself, but we can also make a second observation, which calls into question an unspoken assumption both in the question displayed above and in the idea of maintaining a non-metaphorical E_1 as well as the metaphorical E_3. That assumption is that there *exists* an adequate representational approach in which to couch non-metaphorical renderings (such as E_1) of nested attitude situations in the first place. I claim that this assumption may be false, but I postpone the issue until Section 4.

Another possible objection we must address is the contention that sentence (2) should be given a a literalized interpretation, that is, one that that substitutes a literal description of the situation metaphorically described as a battle. This may seem to many readers a more natural thing to do than to adopt a "metaphorization" of sentence (1). However, we must always remember that the interpretation of (1) and (2) must be such as to allow plausible reasoning about what *Zorn* would (plausibly) infer from her thoughts about Xavier. And this involves reasoning about what Zorn herself would conclude from her metaphorical thought that the idea X is battling the habit of wishful thinking. (Recall that we are assuming an inner-scope interpretation of the metaphor in (2).) Under the literalization-of-(2) proposal, we would have to ensure that the literalization was such that the inferences it supports emulate the inferences Zorn would be likely to make based on her metaphorical thought. I contend that this would be an extremely difficult thing to do, because Zorn might make use of many different aspects of the battleground domain as a basis for her inferences. (The fact that her inferences might themselves be implausible as a result of being based on an ill-advised application of the metaphor is beside the point.) A literalization of (2) would somehow have to encompass many possible ramifications of the metaphor, not just the bare idea that Xavier's habit of wishful thinking is tending to stop Mike's coming to realize that his theory is faulty. In any case, if an adequate literalization were available, it would seem pointless and perverse to use it as opposed to conducting inference at the level Zorn is taken to be thinking, i.e. at the level of the metaphor.

There is a further consideration supporting the claim that the system should conduct its inference about Zorn's thoughts about Xavier at the level of the metaphor, and not move to a literal explication. This consideration is that the major conclusions that Zorn (supposedly) comes to may themselves be at the level of the metaphor, and may be difficult to translate into literal terms. It is natural for a human reader to infer (or at least to accept) on the basis of (1) and (2) that, in Zorn's view, even if Xavier does come to realize that his theory is faulty, the realization is in danger of being destroyed later by the continued onslaughts of his habit of wishful thinking. If the system is later presented with the following input:

(4) *Zorn thinks that even if he does come to realize that his theory is faulty, he'll have to work hard to keep it in mind.*

the system does not to have to argue very far from its internal rendering of (3), and a similarly explicated version of the realizing in (4), to the conclusion that the likely reason for the need for the hard work is the wishful-thinking habit. (An interesting issue that I do not address here is the required mapping between the active MIND-AS-BATTLEGROUND metaphor and the MIND-AS-CONTAINER metaphor brought in by (4).) Indeed, we see that on the basis of (3) and (4) the system could easily derive the plausible conclusion that Zorn thinks that, during the period between the achievement of the realization and the forgetting, the idea that the theory is faulty has been continuously trying to defend itself from attacks by the wishful thinking habit. But this conclusion is one which it would be inefficient or difficult (if not impossible) to express in terms other than of the battleground metaphor (or some other metaphor). It therefore makes sense for the system to conduct its reasoning as far as possible within the terms of that metaphor.

3.2 Sensitivity of Inference to Choice of Metaphor

In the last subsection we discussed the benefit of giving a metaphorical explication to a propositional attitude, such as the realizing in (1). A converse issue arises in examples where what is given is already a metaphorical explication of a mental state, as in

(5) *Zorn believes that Jane has it firmly fixed in her mind that John is evil.*

Assume, again, an inner-scope reading, now with respect to the MIND-AS-CONTAINER metaphor being used to describe Jane's mental state. One might suggest that, even if the system kept around a representation of Zorn as thinking about Jane in terms of this metaphor, the system should use for its main inferencing an (ostensibly) literal paraphrase of the content of the sentence, much as if it had said:

(6) *Zorn believes that Jane believes with high confidence that John is evil.*

I do not deny that there may be good reasons for adopting such a paraphrase. However, I also claim that a MIND-AS-CONTAINER-based internal rendering of (5) is likely to be more useful inferentially. It allows the system to infer, much more easily/securely than (6) does, that, in Zorn's view, Jane could only be made to lose her belief that John is evil with considerable difficulty, and with a lot of resistance from Jane (or from objects in her mind other than the idea that John is evil). This is because the phrase "firmly fixed" makes the clear suggestion that the idea is fixed to *something*: either the container itself, or other ideas, or both. One might respond to this by claiming that (6) should be corrected to say "with high internal justification level" rather than (or as well as) "with high confidence", to capture the idea of the belief being connected to other things. The trouble with such a suggestion is that for every plausible conclusion that one might come to on the basis of the metaphorical idea of "firmly-fixed", one would have to include in (6) a means of capturing it. For instance, it is reasonable to suggest that the system could plausibly conclude from (5) that, in Zorn's view, Jane is rather obstinate: this is because the notion of something being "firmly fixed" naturally, by a process of association, leads one to imagine actions that might dislodge the fixed thing. Any such attempts have (in Zorn's view) failed. The amended version of (6), in not having corresponding dynamic connotations emanating from the commonsense physical world, is as it stands less reasonably claimed to lead to the plausible conclusion that Jane is obstinate. In sum, the metaphorically-explicated internal representation corresponding to (5) is an economical and efficient basis for plausible inferences about what it is that Zorn is likely to conclude about Jane, on the basis, remember, of *Zorn's use* of the MIND-AS-CONTAINER metaphor. Had (5) used a different metaphor, the system would have needed to reason differently about Zorn's reasoning about Jane.

All this is said with the understanding that there are simple types of inference about attitudes that do not require or benefit from metaphorical explication. For instance, if we are told that John gets angry whenever he thinks about musicians, we can infer from the information that he is thinking about musicians direct to the conclusion that he is going to get angry.

There has been much discussion, in philosophy and AI, on what sorts of inference a scheme for propositional-attitude representation/reasoning should force, sanction, or prohibit. In fact, this is *the* strongest unifying thread running through the attitude-representation field. It is the issue underlying the problems of referential opacity and logical omniscience, and most works on propositional attitudes (as an issue in their own right) preferentially address these rather specialized topics. The proposer of a scheme usually makes specific decisions on what inferences to force, sanction or prohibit, and is usually then taken to task by proposers of other schemes. The forcings, sanctions, and prohibitions proposed are very much dependent on the style of representation that the proposer finds appealing (e.g. modal-logic, quotational-logic, neo-Fregean, semantic network), and the appeal depends on what metaphor of mind the proposer finds most useful or congenial.

My own stance is that there is no single right answer to the question of what inferences should be sanctioned, etc., because we have available to us no single correct characterization of mental states (and even if we did it wouldn't be much use because it wouldn't be used by agents holding attitudes about other agents' attitudes), and this is because all we have is bunch of possibly conflicting metaphors for mental states. Therefore, a system that reasons about attitudes will tend to come to different plausible conclusions depending on what types of metaphorical explication it is led to adopt at the time. To give a simple example of the effect I envisage here, consider the sentence

(7) *Mike believes that Jane is tall and that all tall people are clever.*

It is standard practice to wonder whether a person or AI system can conclude from this that Mike believes that Jane is clever. The first point to make is that the issue is one of merely *plausible* inference, whereas most discussions are cast only in terms of sound inference, using the technical, strict notion of "sound". That aside, the important observation is that the answer to the question posed a moment ago is dependent on which metaphor the system hypothesizes the speaker to be using. (The speaker in this example is playing a role comparable to that of Zorn in previous examples.) Suppose, for instance, the system assumes an AGENT'S-WORLD metaphor, and adopts an internal representation paraphrasable as

(7a) *In Mike's world, Jane is tall and all tall people are clever.*

The simpler ways of using the world metaphor within plausible reasoning would lead to the conclusion that in Mike's world Jane is clever — that is, to back-paraphrase, the conclusion that Mike believes that Jane is clever. But suppose now the system uses the MIND-AS-CONTAINER metaphor. To paraphrase, the system has the information:

(7b) *Mike's mind contains the idea of Jane being tall and the idea of all tall people being clever.*

Since we are accustomed to thinking, under the banner of the MIND-AS-CONTAINER metaphor, of ideas being in perhaps widely separated parts of the container, possibly even in "compartments" between which travel is difficult or infrequent, the system must allow the reasonably strong possibility that Mike has not brought the two

ideas referred to in (7b) together in such a way as to facilitate an inference on his part to Jane being clever. This does not of itself mean that the system shouldn't tentatively conclude that Mike does believe that Jane is clever, but it does mean that later evidence contrary to this conclusion can (or should) be used not only to repeal the conclusion, but also to infer that the two ideas are in fact in mutually distant/inaccessible parts of Mike's mind/container. This inference could later be deployed to good effect in the system's reasoning about Mike.

In contrast, in the case in which the system is using the AGENT'S-WORLD metaphor, if it learns contrary to expectation that Mike does not believe that Jane is clever, then, since that conclusion is inescapable under the simple way of deploying that metaphor, the system should either adopt a more sophisticated way of using the metaphor or should abandon it in favor of another one. From this we see that metaphor-based reasoning about propositional attitudes may not only involve defeasibility of inference *within* a metaphor, but also defeasibility of the choice of metaphor in the first place.

3.3 Benefits: Further Discussion

The discourse example (1,2) and the example sentence (5) show that metaphorical explications of attitudes are needed by an AI system as a basis for perceiving the coherence of discourse and for making plausible inferences from discourse. We also saw, using example (7), that the ability to choose a particular metaphorical explication from a range of possible explications allows a system to engage in different patterns of inference about attitudes, in a way guided by the prevailing context. This approach relieves us of the need to commit ourselves to any particular regime of inference forcing, sanctioning and prohibiting. It also elevates the metaphors that have been appealed to by proposers of attitude representation/reasoning schemes, but that are only feebly elaborated in those schemes, to the status of explicitly available representational/reasoning frameworks, endowed with whatever degree of commonsensically-motivated richness is desired.

The considerations supporting the claims are clearest when what is explicitly at issue, in a piece of discourse, is an outer agent's (e.g. Zorn's) view of an inner agent's (e.g. Xavier's and Jane's) attitudes. For instance, we discussed (1), (2) and (5) on the assumption of an inner-scope reading of the metaphors of interest, that is, on the assumption that Zorn herself is thinking in terms of the metaphors. However, the considerations also arise if an AI system is presented with sentence (7), since the attitudes of the speaker of the sentence may well have to be taken into account, and any mental metaphors underlying the speaker's view of Mike are important determinants of the patterns of inference the system can plausibly engage in. But going back now to (1), (2) and (5), we realize that the attitudes of and metaphors used by the speakers are important here too. This not only affects the way the system should interpret Zorn's own attitudes (whereas previously we discussed only the inner attitudes, those of Xavier and Jane in Zorn's view); it also reveals that even on an *outer*-scope reading of the metaphors in (2) and (5) the perception of discourse coherence will need to rest on those metaphors, though in a way differing in detail from the inner-scope case.

4 Avoiding Implausible Ascriptions

In Section 3.1, I advertised a claim that the assumption that there is an existing representational tool that is adequate for the representation of nested attitudes may be false. I do not go so far as to argue that the assumption is definitely false, or that, even if it is false at present it will continue to be false, but I do want to mention that many existing schemes suffer from an "implausible ascription" problem. That is, the schemes in question tend strongly to lead to implausible ascriptions of explicational views of inner attitudes to outer attitude holders. The ascriptions are therefore in contrast to the plausible, metaphor-based ascriptions I base my approach on. I give here only a brief sketch of the problem, because it has been discussed at length elsewhere [Barnden, 1986a, 1987a,b, 1989].

A basic example of the problem arises with typical ways of using quotational logic to represent attitudes. (Analogous problems face situation-based schemes and concept-based schemes.) A quotational-logic approach tends to lead to the ascription, to an outer attitude holder Z, of a view in which *an inner agent being in a certain arcane, highly theory-laden relationship to a formula belonging to the logic itself*. For instance, in the case of sentence (1), the scheme is likely to have the effect of ascribing to Zorn a view of Xavier as being in *a relationship of this sort to a logic formula* expressing the idea that his theory is faulty. Unless Zorn is a propositional attitude researcher it is highly unlikely that she would have such a view of Xavier. What has happened is that the system's own explicatory view of the possession of an attitude as being a matter of a relationship to a logic formula has been implausibly ascribed to real people Z when *they* are taken to represent the possession of an attitude by some agent. Now, it is in fact possible to use the quotational logic, and probably the situation-based and concept-based styles as well, in such a way that no ascription at all of particular ways of explicating attitudes are made to outer agents Z in nested attitude situations [Barnden, 1987a,b]. However, this total avoidance of explicational ascriptions introduces an undesirable degree of representational complexity, such as pervasive, intricate uses of lambda-abstraction or other abstractional tools.

Certain other important approaches, including typical modal logic approaches and the SNePS semantic network approach [Shapiro & Rapaport, 1986], only suffer at most from weak forms of the implausible-ascription problem [Barnden, 1986a, 1987b]. However, modal logic approaches suffer from various problems, some well-known [see e.g. Barnden 1987b, Perlis 1988]; and I have argued elsewhere [Barnden 1986b] that the SNePS system suffers from having no fully general facility for making statements about intensions themselves, despite the fact that intensions are what the nodes represent.

While these observations by no means prove that adequate non-metaphorical representations of nested attitudes cannot be derived, they do show that the task of designing such representations may be much more difficult than is generally realized.

5 Representation Scheme

5.1 Default Representational Approach

The system being developed is called ATT-META. In the absence of any particular contextual cues, ATT-META gives an inner or outer attitude an explication based on an "AGENT'S-WORLD" metaphor. Thus, in the case of the example sentences (1) and (2) above, ATT-META would give a simple, default AGENT'S-WORLD explication to Xavier's hoped-for realization, only to replace it by a BATTLEGROUND explication when the second sentence is processed. Much of the following description shows the default representations that would be used for non-nested attitude situations, in which the only attitude is *a fortiori* outermost.

Under the basic AGENT'S-WORLD metaphor used by default, ATT-META regards an agent's beliefs as defining a "world", which is the agent's version of what ATT-META takes to be the real world. In other words, it is the world as viewed by the agent. Further, ATT-META ascribes, by default, its own world-based way of viewing of agents to the agents themselves. So, the system assumes that, in the world of an agent Z, any agent X has a world; and, continuing the recursion, any agents in X's world as seen by Z themselves have worlds. The world that X has according to Z may of course differ from X's world according to ATT-META.

We need to postulate not only agents' believed worlds, but also their hoped-for worlds, desired worlds, and so on for other types of attitudes; but in the following we discuss only the believed worlds.

Although an agent's world may differ in detail from ATT-META's world (i.e. the "real" world), it is the same *sort* of thing as the latter. It can contain people and other sorts of object, and ordinary sorts of predicate and function apply to those objects. The world idea we are appealing to is therefore akin to the intuitive notion of world in possible-world approaches, but we are not using the notion of *possible* worlds. An agent Z has, in ATT-META's or any other agent's view, just one world — we do not bring in the notion of the set of possible worlds that are consistent with Z's beliefs, which is the standard move in a possible-worlds approach.

A given object can be in more than one world. We simply take this possibility as an unanalyzed presumption of the metaphor we are using. For instance, the real Jane (an entity in ATT-META's world) can also appear in Z's world (according to ATT-META). An entity might only belong to a non-real world, allowing a simple account of fictional entities adequate to the less bizarre ways in which such entities might be reasoned about, although we do not elaborate on this important issue here. Notice the convenience here of taking a metaphor-based view: we are not obliged to ensure that it makes any sort of deep, objective sense to say that an entity can belong to more than one world (an especially striking statement if the entity is fictional). All that we require is that the metaphor involving this assumption is useful in usefully-many practical situations, and is consonant with the way people usefully-often think. Because of differences in belief, Jane can play a different role, have a different physical appearance, have a different name, etc. in Z's world from what she plays and has in ATT-META's world. If the differences are radical enough we might well ask in what sense we can say she is in both worlds. The answer is simply that the metaphor begins to break down if the differences are too radical — if the system notices this, it should adopt another metaphor or a more sophisticated version of the AGENT'S-WORLD metaphor. On the latter option, it could switch to allowing "counterpart" relationships between entities in different worlds, either instead of or as well as allowing an entity to be in several worlds. Counterpart relationships appear in Fauconnier's [1985] "mental spaces" approach (some of his spaces being like our worlds), and in a number of possible world approaches. In the present paper we do not consider counterpart relationships further, but they are included in the preliminary representational proposal of Barnden [1988b, 1989].

One way in which the level of sophistication of the metaphor matters is in allowing more distinctions to be made between interpretations of statements about attitudes. The simple version of the AGENT'S-WORLD metaphor assumed in most of what follows is rather restrictive in the distinctions it allows, but I claim that it is *therefore* adequate as a default approach to many of the simpler cases encountered in realistic contexts.

An interesting complication we must note is that it seems that talk about people's worlds rests on regarding the worlds metaphorically as containers. In a sense we have a WORLD-AS-CONTAINER metaphor sitting on top of, or as part of, the AGENT'S-WORLD metaphor. This does not in fact cause extra complexity in the default representational expressions below, but it leads to

a possible conceptual confusion, because we have the notion of containers being applied in a *different* way in the MIND-AS-CONTAINER metaphor. Under the latter, a container (i.e., a mind) contains beliefs themselves, which are proposition-like ideas (and may be complexes built out of description-like ideas or other proposition-like ideas); whereas under the WORLD-AS-CONTAINER sub-metaphor the container (an agent's world) contains things which the agent's beliefs are *about* (according to that agent) — the beliefs themselves are propositions that hold for those things and are not themselves assumed to be entities in the world. As a special case, of course, the things in the world of agent Z can themselves be ideas. This is because, for instance, Z might adopt a MIND-AS-CONTAINER view of X (replacing the AGENT'S-WORLD view of X that ATT-META ascribes to Z by default). We will deal with this possibility below.

The following description of the default representation is based on showing by example how the scheme would cope with various standard issues of representation, such as distinguishing between de-dicto and de-re readings and dealing with quantification. The examples are entirely cast in terms of belief, but the extension to other types of propositional attitude is straightforward.

We first consider the representation of the information that could be paraphrased in English as

(8) *Mike believes that Jane is clever.*

We explicate this as a matter of "Jane being clever in Mike's world". ATT-META's standard representation of (8) is

(8a) **clever(Jane; W(Mike; sW)).**

In all cases the term following a semicolon relativizes a function or predicate application to a specific world (the one denoted by the term). The constant symbol **sW** denotes the system's world. To simply represent the hypothesis that Jane is clever ATT-META would use the formula **clever(Jane; sW)**. The term **W(Mike; sW)** denotes the world as Mike sees it, according to ATT-META. We will use the abbreviation **MsW** for this term in what follows. The term **W(Mike; W(Bob; sW)**, abbreviated to **MBsW**, denotes the world as Mike sees it, according to Bob (according to ATT-META). For most predicates or functions **R**, an application **R(t; w)** is taken to imply that the entity denoted by **t** is in the world denoted by **w**. Some special predicates and functions do not have this implication.

There is a predicate symbol **in** that allows explicit statements about which worlds entities belong to, and it is used below. This is the same predicate as would be used for physical containment, so that we are using the WORLD-AS-CONTAINER metaphor to project physical predicates directly onto worlds.

Formula (8a) is a sort of de-re interpretation of (8), in that Mike's belief is taken to be about an entity in ATT-META's world (given that **Jane** denotes something in that world), and the role that the entity plays in Mike's world is not constrained by (8a) itself. There is no restriction on substitution of the **Jane** term in (8a), using equality among terms. Assume that ATT-META supposes that **equal(Jane, wife-of(Pete; sW))**. (It does not currently index equality predicates to worlds.) Then it can conclude from (8a) that

(8b) **clever(wife-of(Pete; sW); MsW).**

But this does *not* convey that

(9) *Mike believes that Pete's wife is clever,*

understood de-dicto with respect to "Pete's wife" — that is, understood as implying that Mike is thinking of someone through the wife-of relationship to Pete. Rather, (8b) conveys the very same belief state of Mike as the one (8a) does. Formula (8b) is actually a *de-re* interpretation of (9), that is, taking (9) to mean that Mike believes of Pete's wife that she is clever, with no implication that Mike is thinking of her as Pete's wife.

We should beware that talking about ways in which Mike might be thinking about Jane, Pete and so on is to slip into a different mental metaphor: namely, a metaphor that involves people's beliefs as being composed partly out of "ideas" that may (or may not) succeed in "referring" to things. Formulae (8a) and (8b) should not strictly speaking be discussed in terms of metaphors other than the AGENT'S-WORLD metaphor. Under the latter metaphor, we simply postulate a world (Mike's), containing the very same entity Jane that the *system* knows about through the terms **Jane** and **wife-of(Pete)**, and in which that entity is clever.

The de-dicto interpretation of (9) is

(9a) **clever(wife-of(Pete; MsW); MsW).**

This differs from (8b) only in relativizing the wife relationship to Mike's world. Note that there is again no restriction on substitution, so that if ATT-META supposes that **equal(wife-of(Pete; MsW), Jane)**, then it follows that (8a) holds. From (8a) it could follow by a further substitution that (8b) holds, *as well as* (9a). (It is only when (8b) is in isolation that it is construed as emoodying a de-re reading of (9).) Something else that might follow from (9a), through the formula **equal(wife-of(Pete; MsW), mother-of(Sally; MsW))**, is

(9b) **clever(mother-of(Sally; MsW); MsW).**

That is, the default approach leads to the conclusion that Mike believes that Sally's mother is clever, given that (i)

Mike believes that Pete's wife is clever and that (ii) Mike believes that Pete's wife is Sally's mother (all these belief statements being interpreted de-dicto). This is an instance of the fact that the *default* representational approach forces a form of "logical omniscience" on agents (i.e. they are held to believe all deductive consequences of their beliefs).

Formula (8a) is the standard representation of (8). It is a de-re interpretation, which leaves the question of what a de-dicto interpretation would be. It is not clear from the literature in what sense a proper name can be given a de-dicto interpretation. But there is an alternative, natural interpretation, sometimes called the "metalinguistic" one, under which (8) is rendered as

(8c) **clever(person-called('Jane'; MsW); MsW).**

This is also a de-dicto interpretation for "Mike believes that the person called 'Jane' is clever".

Turning now to the question of quantification, consider

(10) *Mike believes that something is burning.*

The de-dicto interpretation of this, with respect to the quantification, is to convey that in Mike's world there is something or other that is burning (but Mike may not know anything about it other than that it is burning). The interpretation is:

(10a) **(∃x)burning(x; MsW).**

Note that this implies that whatever is burning in Mike's world is, of course, in Mike's world, but it does not imply that it is in the real world. The weak de-re interpretation of (10), under which there is something specific *in the real world* that is burning in Mike's world, but under which there is no commitment as to what role the object plays in Mike's world, is:

(10b) **(∃x)burning(x; MsW) and in(x, sW; sW).**

A stronger type of de-re reading results from assuming that the object does have some definite role to play in Mike's world, such as being Jane's house (although the role is not assumed to be known to ATT-META). We can get this effect by adding the conjunct **plays-specific-role-in(x, MsW; sW)** to (10b). Such a conjunct could also be added to strengthen the weak de-re reading (8b) of (9). Some researchers demand that a de-re reading be strong to the extent of involving the assumption that the object is in some sense directly or vividly known to the agent (Mike). We could get this effect with an extra conjunct like **vivid-to(x, Mike, MsW; sW).**

A less often discussed type of reading of (10) that is in some sense a compromise between de-re and de-dicto takes the burning thing to play a specific though unknown-to-the-system role in Mike's world, and does not assume that the thing is also in the real world. This interpretation can be constructed by adding **plays-specific-role-in(x, MsW; sW)** to the de-dicto interpretation (10a).

We turn now to various interpretations of the attitude statement

(11) *Bob believes that Mike believes that Pete's wife is clever.*

The trouble with this is that there is, so to speak, a de-dicto/de-re choice of interpretation of "Pete's wife" at each attitude level. To keep things simple, we consider here only a choice between de-dicto and weak de-re, and take the straightforward, de-re approach to proper names that is exemplified by the **Jane** in (8a). What we can call the "de-dicto × de-dicto" reading is as follows, where the term **MBsW** is an abbreviation for the term **W(Mike; W(Bob; sW))**, which denotes Mike's world according to Bob.

(11a) **clever(wife-of(Pete; MBsW); MBsW).**

That is, in Mike's world according to Bob, the wife-in-Mike's-world-according-to-Bob of Pete is clever.

The "de-dicto × de-re" reading is as follows, where **BsW** is an abbreviation for **W(Bob; sW).**

(11b) **clever(wife-of(Pete; BsW); MBsW).**

The difference from (11a) is that Bob is now taken to believe that the person in his own world who is Pete's wife is believed by Mike to be clever, there being no assumption on Bob's part that the person he takes to be believed by Mike to be clever plays the role of Pete's wife in Mike's world (according to Bob). Notice that by putting ATT-META in Bob's shoes — by which I mean: replacing **MBsW** by **MsW** and **BsW** by **sW** in (11a) and (11b) — we get, respectively, the de-dicto and de-re interpretations (9a) and (8b) of (9).

Continuing with our combinatorial mini-explosion of readings, the "de-re × de-re" reading is:

(11c) **clever(wife-of(Pete; sW); MBsW).**

That is, the person who is Pete's wife in ATT-META's world is believed by Bob to be believed by Mike to be clever; there is no specification of roles that the entity plays in Bob's world or in Mike's world according to Bob.

The most difficult reading to handle and to explain is the "de-re × de-dicto" one. Under this interpretation, ATT-META takes Bob to believe that there is someone (in Bob's world) whom Mike believes to be clever, where the someone is specified by Bob to play, in Mike's world, a specific role (such as being Sally's mother) that the person also plays in Bob's world; where, however, ATT-META does not know what that role is, and only knows that the someone is Pete's wife in the real world. The closest I have come so far to a satisfactory formulation of this in the default representational approach is:

(11d) **clever(wife-of(Pete; sW); MBsW) and**
plays-some-same-role-in
(wife-of(Pete; sW), BsW, MBsW; sW).

The special conjunct here specifies that the person known to ATT-META as Pete's wife plays, according to ATT-META, some role in world **MBsW** that is the same as the role played in world **BsW**. A better treatment of such readings may require reifying roles to make them become entities in worlds, and to allow a role to be in several worlds and/or to allow counterpart relationships among roles in different worlds [Fauconnier, 1985].

5.2 A MIND-AS-CONTAINER Representation

Here I briefly explain how ATT-META would represent single-level attitudes on the basis of a MIND-AS-CONTAINER metaphor. (A MIND-AS-BATTLEGROUND representation would be similar, but would allow representation of battle relationships among the ideas, etc. A minor complication is that the MIND-AS-BATTLEGROUND metaphor is really a metaphor about the contents of a mind that is already subject to a MIND-AS-CONTAINER view.) The MIND-AS-CONTAINER representations are akin to those used in a number of representational approaches that have been proposed (mainly, intensional logics and neo-Fregean schemes). The following formula would be used to represent the piece of information (9) under a de-dicto interpretation:

(9c) **in(ι(clever(wife-of(Pete))), mind-of(Mike)).**

The ι is an operator used to form terms that denote ideas. The ι term in (9c) denotes the idea which could be paraphrased in English by the sentence "the wife of Pete is clever". The operator sets up an intensional scope, which means among other things that substitution of equals for equals within the scope changes the denotation of the term.

Notice that we are using some of the same predicate and function symbols as were used above, but are dropping the world-relativization term. This reflects a deliberate departure from ordinary logic. We allow different applications of a predicate or function symbol to have different numbers of arguments, and in general we allow a keyword-based form of argument specification so as to avoid ambiguities. We could instead have insisted upon using different predicate/function symbols, but this would have made the total approach unnecessarily complex.

Note also that the **in** symbol in (9c) still denotes physical containment. This reflects the fact that both minds and worlds are metaphorized as containers.

To cope with readings of (9) where we have no full specification of the way Mike is thinking of the clever person — as in most types of de-re reading — we use a variant of the ι operator as in the formula:

(9d) **(∃i)in(ι_1(clever(1:i)), mind-of(Mike))**
and an-idea-of(i, wife-of(Pete)).

This is the weak de-re interpretation of (9). For any given hypothetical assignment of an idea I as a denotation to the variable i, the ι_1 term here denotes the idea obtained by taking the idea C of "somebody-or-other is clever" and plugging I into C to replace the "somebody-or-other" idea. (This is only well-defined if I is a term-like idea as opposed to a proposition-like idea.) This rough account could be fleshed out in a formal semantics in terms of a suitable ontology of structured ideas. Using the ι operators it is straightforward to reformulate, in MIND-AS-CONTAINER terms, the various AGENT'S-WORLD-based readings of examples (8) to (11) above.

5.3 Mixed Representation

What is of most interest for us here is the use of one metaphor to explicate the attitude at one level in a nested-attitude situation, and another metaphor to explicate the attitude at another level. For instance, context might suggest that the inner attitude in (11) should be given a MIND-AS-CONTAINER explication, although there is nothing to indicate that the outer one should not be given the simple AGENT'S-WORLD explication. We need to augment the representation in the previous sub-section by adding world-denoting terms to the **in**, **mind-of** and ι applications. Then, for a de-dicto × de-dicto reading of (11) we get:

(11e) **in(ι(clever(wife-of(Pete)); BsW),**
mind-of(Mike; BsW); BsW).

We do not world-relativize things inside the ι term's first argument, since that term denotes an idea in Bob's world: and *Bob* is not now being taken to have a world-based view of agents. What (11e) says is that the idea-in-Bob's-world of Pete's wife being clever is, from the

point of view of Bob's world, in Mike's mind. A de-dicto × de-re interpretation of (11) is:

(11f) $(\exists i)$**in**(ι_1**(clever(1:i); BsW),**
mind-of(Mike; BsW); BsW)
and an-idea-of(i, wife-of(Pete; BsW); BsW).

A de-re × de-re interpretation is obtained from this simply by replacing the **BsW** term in the **wife-of** application by **sW**. A de-re × de-re interpretation is obtained from (11f) by replacing the **an-idea-of** conjunct by the conjunct

is-a-descriptional-idea-for
(i, wife-of(Pete; sW), BsW; sW).

This states that the idea i serves to describe, in the context of Bob's world, Pete's-wife-in-the-system's-world.

It is also possible to construct mixed representations where the inner one is based on the AGENT'S-WORLD metaphor and the outer one is based on the MIND-AS-CONTAINER metaphor. These are omitted for the sake of brevity.

6 Concluding Remarks

I propose that attitudes be explicated in terms of commonsense, and largely metaphorical, views of mind that are held by people and that are prevalently expressed in natural language discourse about attitudes. A formal representation scheme is being developed that allows these explications to be expressed. The system has available to it a variety of possible explications for attitudes, and the choice amongst them is, in the natural language processing case, guided by considerations of discourse coherence. In the absence of contextual cues suggesting a particular explication, a default explication based on a simple AGENT'S-WORLD metaphor is used. In the representation of a nested attitude situation, different metaphors can be used for explicating the attitudes at different levels.

The representational expressions arising from the default, AGENT'S-WORLD explications look much like the expressions in some *existing* attitude representation schemes based on notions of space, environment or partition. Therefore, not much extra complexity is being caused in straightforward cases by our wholesale application of the idea that attitudes be commonsensically explicated. However, our use of AGENT'S-WORLD representations does constitute a departure from existing space-based schemes, as it involves an explicit commitment to taking the represented agents themselves to be using the world metaphor.

The approach allows a more realistic and powerful attack on propositional attitudes. Part of the power results from removing the obligation on us to ensure that any *single* way of representing attitudes is capable of providing the right sanctions and prohibitions of various types of inference (such as inference by term substitution). Rather, different patterns of inference are viewed as being tied to different metaphors. For instance, a disadvantage of the AGENT'S-WORLD representations is that they force a type of logical omniscience. In many realistic cases this may not cause problems; however, if the resulting inferences prove in context to be incorrect, a more sophisticated form of the metaphor, or a different metaphor, must be used. For instance, the MIND as CONTAINER or as BATTLEGROUND metaphors, which have very little power to force particular types of inference about an agent's beliefs, could be adopted.

The proposed approach makes rich contact with recent work in linguistics and philosophy on commonsense views of mind, such as [Lakoff, 1987, Johnson, 1987, Sweetser, forthcoming]. It emphasizes the strong dependence of a proper treatment of propositional attitudes on plausible as opposed to definite reasoning [Barnden, 1988a]. It supports the idea that attitude verbs and the like in natural language are highly polysemous, vague terms. Naturally, major issues remain to be attacked in detail. The formal representation needs to be extended to metaphors other than AGENT'S-WORLD, MIND-AS-CONTAINER and MIND-AS-BATTLEGROUND. Some systematic way of using different metaphors at different levels should be developed. The approach should be seen as a special case of two more general issues [Barnden, 1988a,b]: of metaphors (in general) in attitude contexts (consider "Mike believes that Jane is a *snake*"); and of vague/polysemous terms (in general) in attitude contexts (consider "The Admiral believes that his ship is *threatened*"). Finally, the question of how particular metaphors are selected and elaborated according to context is a major area of research.

Acknowledgments

I have benefited from suggestions by Jerry Ball, Afzal Ballim, David Farwell, Steve Helmreich, Mark Johnson, Paul McKevitt and Yorick Wilks.

References

Barnden, J.A. (1986a). Imputations and explications: representational problems in treatments of propositional attitudes. *Cognitive Science, 10* (3), 319-364.

Barnden, J.A. (1986b). A viewpoint distinction in the representation of propositional attitudes. In *Proceedings of the 5th Nat. Conf. on Artificial Intelligence (AAAI86)*, Philadelphia, Penn.

Barnden, J.A. (1987a). Interpreting propositional attitude reports: towards greater freedom and control. In B. du Boulay, D. Hogg & L. Steels (Eds),

Advances in artificial intelligence - II, Amsterdam.: Elsevier (North-Holland).

Barnden, J.A. (1987b). Avoiding some unwarranted entailments among nested attitude reports. *Memoranda in Computer and Cognitive Science*, No. MCCS-87-113, Computing Research Laboratory, New Mexico State University, NM 88003, USA.

Barnden, J.A. (1988a). Propositional attitudes, commonsense reasoning, and metaphor. In *Procs. 10th Conference of the Cognitive Science Society*, Hillsdale, N.J.: Lawrence Erlbaum.

Barnden, J.A. (1988b). Propositional attitudes, polysemy and metaphor. *Memoranda in Computer and Cognitive Science*, No. MCCS-88-139, Computing Research Laboratory, New Mexico State University, NM 88003, USA.

Barnden, J.A. (1989). A misleading problem reduction in belief representation research: some methodological considerations. *J. Experimental and Theoretical Artificial Intelligence, 1* (2), 4-30.

Black, M. (1954/5). Metaphor. *Procs. of the Aristotelian Society*, n.s. 55 (1954-5), 273-294.

Carbonell, J.G. (1982). Metaphor: an inescapable phenomenon in natural-language comprehension. In W. Lehnert & M. Ringle (eds), *Strategies for natural language processing*, Hillsdale, N.J.: Lawrence Erlbaum.

Dinsmore, J. (1988). Partitioning knowledge as a fundamental process in understanding natural language discourse. Tech. Rep. No. 88-25, Dept. of Computer Sci., Southern Illinois University at Carbondale, Carbondale, Ill., USA.

Fass, D.C. (1987). Semantic relations, metonymy, and lexical ambiguity resolution: a coherence-based account. In *Procs. of the 9th Annual Conference of the Cognitive Science Society*, Hillsdale, N.J.: Lawrence Erlbaum.

Fauconnier, G. (1985). *Mental spaces: aspects of meaning construction in natural language*. Cambridge, Mass.: MIT Press.

Hobbs, J.R. (1983a). Metaphor interpretation as selective inferencing: cognitive processes in understanding metaphor (Part 1). *Empirical Studies of the Arts, 1* (1), 17-33.

Hobbs, J.R. (1983b). Metaphor interpretation as selective inferencing: cognitive processes in understanding metaphor (Part 2). *Empirical Studies of the Arts, 1* (2), 125-141.

Hobbs, J.R. (1985). On the coherence and structure of discourse. Report No. CSLI-85-37, Center for the Study of Language and Information, Stanford University, Calif., USA.

Johnson, M. (1987). *The body in the mind*. Chicago: Chicago University Press.

Katz, A.N., Paivio, A., Marschark, M. & Clark, J.M. (1988). Norms for 204 literary and 260 nonliterary metaphors on 10 psychological dimensions. *Metaphor and Symbolic Activity, 3* (4), 191-214.

Lakoff, G. (1987). *Women, fire and dangerous things*. Chicago: University of Chicago Press.

Lakoff, G. & Johnson, M. (1980). *Metaphors we live by*. Chicago: University of Chicago Press.

Larsen, S.F. (1987). Remembering and the archaeology metaphor. *Metaphor and Symbolic Activity, 2* (3), 187-199.

Perlis, D. (1988). Languages with self-reference II: knowledge, belief, and modality. *Artificial Intelligence, 34* (2), 179-212.

Reddy, M.J. (1979). The conduit metaphor — a case of frame conflict in our language about language. In A. Ortony (Ed.), *Metaphor and Thought*, Cambridge, UK: Cambridge University Press.

Richards, I.A. (1936). *The philosophy of rhetoric*, Oxford: Oxford University Press.

Shapiro, S.C. & Rapaport, W.J. (1986). SNePS considered as a fully intensional propositional semantic network. *Procs. 5th National Conf. on Artificial Intelligence (AAAI-86)*, Los Altos, CA: Morgan Kaufmann.

Sweetser, E.E. (1987). Metaphorical models of thought and speech: a comparison of historical directions and metaphorical mappings in the two domains. In L. Michaelis, J. Aske & H. Filip (Eds), *Procs. 13th Annual Meeting of the Berkeley Linguistics Society*.

Sweetser, E.E. (forthcoming). *From etymology to pragmatics*, Cambridge: Cambridge University Press.

Talmy, L. (1988). Force dynamics in language and cognition. *Cognitive Science, 12*, 49-100.

Tomlinson, B. (1986). Cooking, mining, gardening, hunting: metaphorical stories writers tell about their composing processes. *Metaphor and Symbolic Activity, 1* (1), 57-79.

Wilks, Y. & Ballim, A. (1987). Multiple agents and the heuristic ascription of belief. *Procs. 10th Int. Joint Conf. on Artificial Intelligence*, Los Altos, CA: Morgan Kaufmann.

Wilks, Y. & Bien, J. (1983). Beliefs, points of view and multiple environments. *Cognitive Science, 8*, 120-146.

Hierarchical Knowledge Bases and Efficient Disjunctive Reasoning

Alex Borgida
Department of Computer Science
Rutgers University
New Brunswick, NJ

David W. Etherington
AI Principles Research Department
AT&T Bell Laboratories
Murray Hill, NJ

Abstract

We combine ideas from relation-based data management with class hierarchies to obtain Hierarchical Knowledge Bases, which have greater expressive power while maintaining the benefits of predictable and efficient information processing. We then consider the problem of reasoning with certain limited forms of disjunctive information. We show that hierarchical knowledge bases can be used for efficient *approximate reasoning* with such information.

The significant features of our approach include a well-conditioned trade between efficiency and accuracy, with a sound and complete limit case, and approximations guided by the structure of the domain theory. Because of the structure imposed on the knowledge base, it is possible to characterize the potential error in any approximation.

1 Introduction

It is fashionable to view a knowledge base (KB) as an integral utility invoked by a problem-solving program. To be useful in tasks such as robot guidance, such a KB subsystem must perform efficiently, and do so *predictably*. Systems that process sensory input, or "computer individuals" with extended "lifetimes" [Nilsson, 1983], must store and retrieve vast quantities of data. For such systems to respond in real time, the performance of their KBs must not degrade noticeably as new information is acquired.

Ideally, a KB would perform like a relational database—simple queries and updates would be done in time sub-linear in the size of the KB; even more complex queries should take no more than polynomial time. Unfortunately, relational databases are notoriously lacking in expressive power: there are many situations, usually involving partial information and rules, that they cannot describe. Logic-based KBs do not suffer from these defects, but unfortunately are generally undecidable, or at least intractable.

Several approaches to this problem have been considered:

- *Heuristics:* Although they can provide significant performance gains, heuristics tend to be domain dependent, and not to perform predictably.
- *Restricted KB interaction languages:* These restrict the language of KB interactions to frequently-occurring, or particularly useful, sublanguages, in the hope of gaining efficiency. Horn theories are a successful example of this approach. Unfortunately, as we shall describe, disjunctive information has not yielded to this technique.
- *Approximate reasoning:* For example, Imielinski [Imielinski, 1986, Imielinski, 1987], Plaisted [Plaisted, 1981], and others have considered reasoning in a more *abstract* language. A different approach, suggested by Hector Levesque, that we have pursued with Ron Brachman, Henry Kautz, and Bart Selman, has focused on transforming incomplete information into a *vivid* form, where complex reasoning is avoided through various techniques such as defaults/preferences, arbitrary choices, etc [Etherington *et al*, 1989].

In this paper we apply a combination of these techniques. *Limited languages:* We take the ubiquity of class hierarchies in AI knowledge representation schemes, and the success of relational database systems, as license to define Hierarchical Knowledge Bases (HKBs), which combine the best features of each. We identify subclasses of naturally-occurring disjunctive sentences (which we call "uniform disjunctions"), and show how HKBs can be used to implicitly represent and reason with such disjunctions. *Approximation:* We show that by carefully tailoring the hierarchies in HKBs, we get approximate, but provably-efficient, reasoning with certain disjunctive sentences. *Heuristics:* We argue that the hierarchies and disjunctions that are likely to occur in practice will lead to reasonable and useful approximations.

The plan of the paper is as follows: Section 2 describes some of the underlying formal reasons for the efficacy of relation-based and taxonomic knowledge representation schemes. We then present data

structures and algorithms for HKBs that combine these features effectively. Thereafter, we examine two useful families of disjunctive sentences: *uniform-argument clauses* where all disjuncts share the same argument(s) (e.g., $Dog(fuzzy) \vee Cat(fuzzy)$), and *uniform-predicate clauses* where all disjuncts share the same predicate and all but one argument (e.g., $Visit(ron, toronto) \vee Visit(ron, montreal)$). We consider these kinds of sentences more typical of commonsense reasoning than more varied disjunctions (e.g., $Doctor(joe) \vee Cat(fuzzy)$).

In each case we proceed as follows:

- show, using results from complexity theory, that it is impossible to obtain the desired performance even within these limited subclasses of formulae;
- show that reasoning with such sets of clauses, or formulae that are equivalent to them, can be simulated by HKBs with appropriate, but very large, hierarchies;[1]
- show how the process of updating and querying the KB can use a (possibly much smaller) *sub*hierarchy to provide *approximate reasoning*, where the amount of vagueness (or even error) varies depending on the queries.

2 Hierarchical Knowledge Bases (HKBs)

2.1 Motivation

Our approach to efficient reasoning elaborates a straightforward marriage of inheritance hierarchies and relational databases (RDBs), combining some of the best of both.

Relational databases have a number of advantages, including their close and clean relationship to logic, their uncontroversial theory of destructive updates, and the existence of efficient data structures (for example, AVL trees, which give logarithmic-time insertion and search operations for atomic facts). From the logical point of view [Reiter, 1984], the advantages of RDBs are the result of decisions to:

- limit the language of assertions (updates) to positive atomic formulae;
- represent negative information only implicitly, (and thus save a great deal of space) by making the Closed-World Assumption (CWA), which asserts that a fact is false unless explicitly asserted;
- adopt the unique-name and domain-closure assumptions (UNA and DCA), which assert that the domain of discourse includes precisely the individuals explicitly mentioned in the KB, to avoid the need to reason about the identity of objects.

[1] Of course, the complexity results continue to hold, so this transformation, in itself, is not much use.

Unfortunately, the exclusion of incomplete information makes RDBs poorly-suited to many applications, where vague, missing, or disjunctive information occurs. Efforts to extend database technology to allow even limited forms of incomplete information (such as *null values* [Codd, 1979]) have been fraught with difficulty and have—thus far—failed to achieve the desired performance [Imielinski, 1988].

Inheritance hierarchies provide representational capabilities somewhat orthogonal to those provided by RDBs. They can be used to express and enforce inclusion constraints, such as *All P's are Q's*, allowing the inference of facts from others. They also allow for some types of incomplete information: for example, if P's and R's are Q's, it is possible to tell the network $Q(a)$ without committing to the truth values of either $P(a)$ or $R(a)$. If we limit ourselves to hierarchies without "inheritance cancellation", reasoning with hierarchies can also be fast [Agrawal *et al*, 1989].

Because strict inheritance reasoning can be done quickly, some researchers have been led to create "hybrid" reasoning systems, in which a separate inheritance reasoner augments a more general problem-solver, to exploit the tractability of subproblems that involve only inheritance reasoning [Brachman *et al* 1985]. A complimentary approach, which we pursue here, is to use semantic network representation structures in concert with database technology to obtain efficiency and expressive power currently available in neither.

2.2 Definition of HKBs

The class hierarchy is a directed, acyclic, graph over the predicates of the language.[2] A node may have more than one parent, but there is no notion of inheritance cancellation. The presence of a link from node P to parent node Q expresses, as usual, the logical implication $\forall x.\ P(x) \supset Q(x)$. For now, we assume that all non-primitive classes (i.e., non-leaf predicates) in the hierarchy are equivalent to the union of the classes they subsume (e.g., $Mammal$ is exhausted by its subconcepts, Dog, Cat, etc.). This comprises a form of *intensional* closure, corresponding to Clark's [1978] "Predicate Completion". Logically, this amounts to the assumption that non-primitive predicates are equivalent to the disjunction of their immediate children.

The remaining closure assumptions are *extensional*, and are made at query-evaluation time. We continue to make the DCA and UNA, but are forced to adopt a more careful version of the closed-world assumption, Minker's [1982] "generalized closed-world assumption" (GCWA), to avoid inconsistencies. The GCWA supports inferring the negation of a ground literal provided it is false in all *minimal models* of the knowledge base. This amounts to assuming the KB has

[2] We use unary predicates for brevity, but the arguments may be viewed as vectors.

complete information except where it explicitly contains incomplete information. Thus, if *Mammal* is specified to have subclasses *Dog* and *Cat*, then having been told *Dog(joe)*, the KB will conclude $\neg Cat(joe)$; on the other hand, knowing only *Mammal(Joe)*, the KB can conclude neither $\neg Cat(joe)$ nor $\neg Dog(joe)$.

We view a KB system as defined completely in terms of its behavior at its *TELL* and *ASK* interfaces [Levesque, 1984]:

$TELL : L_{TELL} \times KB \rightarrow KB$

$ASK : L_{ASK} \times KB \rightarrow \{\textbf{Yes}, \textbf{No}, \textbf{Unknown}\}$

We take L_{TELL} and L_{ASK}, the languages of expressions in which the KB accepts assertions and queries, respectively, to be subsets of function-free FOL. An HKB is then *specified* to behave as a first-order theory (consisting of the axioms implied by the taxonomy, and the formulae told to the KB), augmented by the DCA, UNA, and GCWA. A question should be answered **"Yes"** if it is entailed by the above, **"No"** if its negation is entailed, and **"Unknown"** otherwise.

L_{TELL} and L_{ASK} can be varied to create a range of update and query problems, with an attendant range of complexities. Since the GCWA provides a form of negation, we restrict ourselves here to languages without negation: these suffice to provide a wide range of reasoning complexities, and are practically useful (c.f., PROLOG).

The following sections consider data structures and algorithms for updating and querying HKBs in a manner similar to relational databases.

2.3 Data Structures and Algorithms

There are two main data structures. The first represents the hierarchical relationship between the class names.[3] The second structure is an RDB containing a unary relation for each node in the class hierarchy and a binary relation for each primitive predicate.[4] The first relation lists individuals explicitly asserted to belong to the class associated with the node. The second relation maps individuals to an associated counter (UNKNOWN-counter), used to keep track of the ambiguity in the original information. Intuitively, this counter will be non-zero for an individual, b, in the primitive relation, P, if and only if $ASK(KB, P(b))$ is neither **"Yes"** nor **"No"** for the current KB.

Consider now the simplest case, that of *TELL*ing the KB only ground atomic facts. In this case, the algorithm records facts as low as possible in the hierarchy. This ensures that more specific information takes precedence over vaguer information, in accord with the GCWA. In particular, the *TELL* algorithm treats a fact, $P(a)$, as follows. If P, or any node in the subgraph below P, is marked true for a, the *TELL* is redundant, since it is already implied by the KB, and is ignored. Otherwise, if P is subsumed by (i.e., below) any node(s) marked true for a, the *TELL* refines previous knowledge: Unmark each such node for a, and decrement a's UNKNOWN-counter for each leaf node below any such node. Finally mark P true for a, and increment the UNKNOWN-counter for a for each leaf below P.

Querying the KB is slightly more complicated than querying RDBs and inheritance networks, due to the possible presence of incomplete information. Affirmative answers to queries are entailed by the KB assertions; negative answers follow from the CWA in cases where the KB has no information; but the system must be prepared to answer **"unknown"** when the available information is incomplete. A query may be answered unknown if it is not known to be true and it is either entailed by another query whose answer is **"unknown"**, or it is an alternative in a disjunction that is either true or itself unknown.

The UNKNOWN-counter flags set at leaves during *TELL* make it unnecessary to look both up *and* down in the hierarchy before returning an answer. To answer an atomic query, $P(a)$, the system queries the unary relations associated with P and its descendants, breadth first, to see if a is in any of their extensions. If so, the answer is **Yes**. If not, the system inspects the associated binary relation at each leaf below P for a positive UNKNOWN-counter for a. If one is found, it returns **Unknown**. Otherwise, **No** is returned.

For example, given the hierarchy in figure 1,[5] asserting *Rodent(stanley)* will result in Rodent, Mammal and Pet being answered **yes** for stanley; Mouse, Gerbil, and Hamster being **unknown**, and everything else being answered **no**. The subsequent assertion that stanley was a Hamster, which refines the previous assertion, would make the answer to Mouse and Gerbil **no**, and to Hamster **yes**.

More general queries could be handled according to the standard truth-functional decomposition used in relational databases: reducing the query to the truth value of component atomic formulae. This approach is however incomplete: If we are asked $P_1(a) \vee ... \vee P_n(a)$, and there is a node Q in the hierarchy that subsumes exactly nodes $P_1, ..., P_n$ then the answer should be the same as that for querying $Q(a)$, rather than combining the truth values of independently querying each $P_i(a)$. If no node in the hierarchy corresponds to the queried disjunction, the proper way to answer it is to consider all the maximal nodes in the hierarchy that correspond to parts of the query (i.e., are subsumed by the query), query them, and then return the disjunction of the results.

[3] To aid efficiency, each node maintains pointers to the leaves it dominates.

[4] Remember that, while for simplicity of exposition we deal only with unary predicates, additional columns would cause no special problems.

[5] The structure of the hierarchies presented throughout should not be construed as an ontological commitment to the world-view they imply!

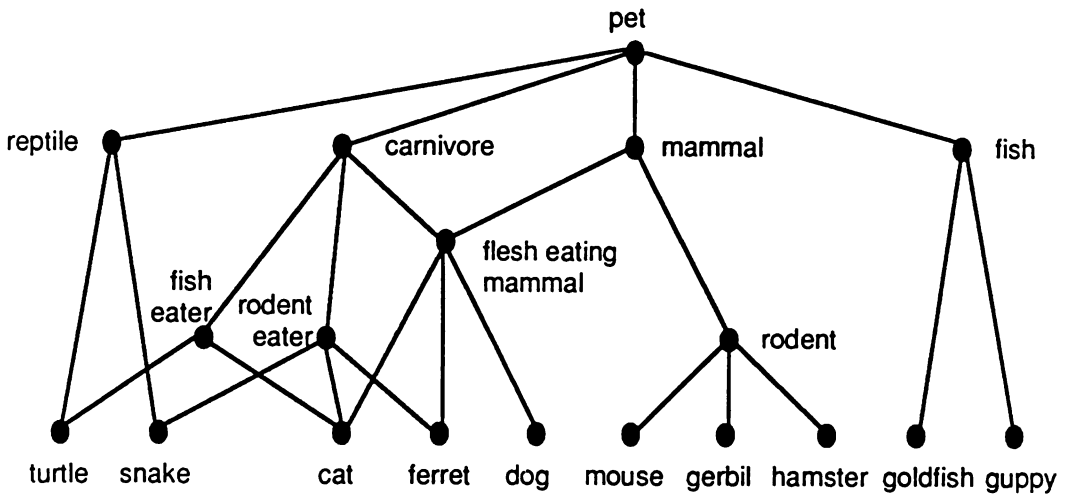


Figure 1: A simple pet-shop world view

Thus, using figure 1's hierarchy, the correct answer to $Mouse(Stan) \vee Gerbil(Stan) \vee Hamster(Stan)$ should be obtained by querying $Rodent(Stan)$, while the best answer available to $Turtle(sid) \vee Snake(sid) \vee Fish(sid)$ is the disjunction of the answers to $Reptile(sid)$ and $Fish(sid)$.

Non-clausal ground queries can be handled by converting them to conjunctive normal form (CNF) and then querying each clause separately. Universal and existential queries are answered by eliminating the quantifiers through the use of the DCA, which guarantees a finite number of substitutions in the absence of function symbols.

The following theorem summarizes the complexity and correctness properties of the above algorithms.

Theorem

Suppose there are n nodes in the hierarchy, and L_{TELL} is restricted to atomic facts, at most m of which have been told to the KB.

- The above algorithms are sound and complete with respect to the specification in the case where L_{ASK} consists of the function- and negation-free subset of FOL.
- If L_{TELL} and L_{ASK} consist of ground atomic formulae, worst-case update and query complexity are $O(n * \log m)$.
- For L_{ASK} = uniform clauses, query complexity is $O(k * \log k * n * \log m)$ where k is the length of the query. (Note that n could be exponential in k, since there are exponentially-many uniform disjunctions as a function of the number primitive predicates.)
- For L_{ASK} = arbitrary ground formulae, the conversion of the query to CNF may increase the length of the query, hence it processing cost, exponentially.
- For L_{ASK} = arbitrary formulae, universal and existential quantifiers may increase the complexity by $O(d^k)$, where d is the size of the domain, and k is the length of the query. ■

The *TELL* and *ASK* algorithms given above extend RDBs in two ways. They provide a mechanism for enforcing "inclusion" constraints (e.g., $\forall x.\ P(x) \supset Q(x)$), and they allow the possibility that certain queries may be answered **Unknown**, even in the presence of closed-world assumptions. Obviously, the complexity of question-answering also somewhat exceeds that for RDBs; for atomic facts this difference is dependent entirely on the size of the hierarchy. As we shall see, though, our algorithm does as well as could be expected, given the inherent computational complexity associated with the increased expressiveness. In the following sections, we consider using this framework to extend the expressiveness of the system still further—in particular, we expand the *TELL* language.

3 Unique-Argument Uniform Clauses

3.1 The General Theory

The previous section considered the problem of querying, using variously restricted languages, an HKB that has been told only ground atomic facts. This section considers *TELL*ing the KB certain uniform disjunctions—specifically, quantifier-free ground clauses of the form $(P_1(a) \vee ... \vee P_n(a))$.

Before proposing ways to deal with such clauses using HKBs, it is worth considering the intrinsic complexity of the problem—how well could *any* algorithm

deal with this issue? The following results from complexity theory are relevant.

An upper bound: We know that if $L_{ASK} = L_{TELL}$ = unique-argument uniform clauses, answers can be obtained in time linear in the total size of the formulae told a KB: it suffices to check whether the query clause subsumes (has as atoms all the atoms of) one of the clauses told the KB. In the worst case, this approach may require looking through all the clauses told the KB, and hence falls short of the desired guarantee of sub-linear response time. In fact, we cannot do better using some other clever technique:

A lower bound: Results from monotone circuit complexity show that if L_{TELL} uses p predicate symbols, then the circuit, and hence Turing complexity of answering even atomic questions is exponential in p [Pippinger, 1978, Pippinger & Fischer, 1979]. (Note: there are exponentially-many uniform clauses in p predicate symbols and one constant; so describing a circuit may require telling the KB very many clauses!) Hence there can be no correct query-processing algorithm for atomic queries that is sub-linear in the number of facts told the KB.

Another lower bound: If we extend the query language, L_{ASK}, to include positive ground formulae with both conjunction and disjunction, the complexity of answering queries is *co-NP-hard* in both the size of the query and the size of the KB. (This result is proved by reduction from the unsatisfiability of 3-CNF.)

Essentially, these results indicate that telling the KB uniform disjunctions makes even the task of answering simple atomic queries more difficult, and makes the task of answering negation-free ground queries intractable. We remark that these results are independent of the decision to make some sort of closed-world assumption.

Given these general complexity results, we cannot hope to do better by encoding the facts in hierarchical KBs (or any other representation). Such an encoding, however, suggests a notion of *approximation* that is not so readily visible in the ordinary logical notation.

3.2 Encoding Clauses in Hierarchies

One can represent uniform clauses over a set of primitive predicate symbols, $P_1, ..., P_n$, by creating a HKB with a node for every clause obtainable from this set—the complete lattice of subsets. A clause, $P(a) \vee Q(a)$, told to the HKB is transformed into $P\text{-}or\text{-}Q(a)$ and represented directly by the HKB. Clausal queries are similarly transformed, and the question-answering algorithm of section 2.2 is invoked. More complex quantifier-free queries are first put into clausal form, then answered clause by clause.

Theorem

The above procedure yields sound and complete answers with respect to the specification. The complexity of answering various queries is at the bounds given in section 3.1. ■

3.3 Approximation Using HKBs

The above shows that HKBs—used as a "theorem proving" mechanism for uniform clauses—perform as well as we currently can expect. It would be easy to dismiss the previous section, if the story ended there. An elaborate representation scheme is presented, based on an unrealistic notion of a complete lattice of predicates, that restricts the logical form of information that can be represented, all in aid of achieving an algorithm with no particular complexity improvements! Fortunately, the story does not end there.

Far from the exponential size of the complete hierarchies discussed above, AI representation schemes more commonly have sparse hierarchies whose size grows only linearly in the number of primitive predicates available to the system. The connections between predicates in such hierarchies are much less profligate, motivated by domain analysis, and generally reflecting observed connections assumed to be useful to the reasoning system. By relaxing the requirement for a complete hierarchy, tending instead toward the sparse hierarchies familiar in AI, significant tractability gains become possible.

For example, given the primitive predicates in figure 1 (Turtle, Snake, Cat, Ferret, Dog, Mouse, Gerbil, Hamster, Goldfish, and Guppy), instead of having classes for all possible subsets of them (1023 classes), we might have just the 9 additional classes shown in the figure (e.g., Rodent (= $Mouse \vee Gerbil \vee Mole$)).

3.3.1 Approximating Uniform Disjunctions

A uniform disjunction, $P_1(a) \vee ... \vee P_n(a)$ can be approximated in a sparse hierarchy as follows. Find the lowest node(s), P, in the hierarchy that dominates all the P_i and assert $P(a)$. If this node is equivalent to the given disjunction, the representation is complete. If not, only an approximation can be represented. This leads to loss of information, but notice that this loss is motivated by the terminology of the domain—forgetting is not just arbitrary.

Since our HKB algorithms for answering positive clause queries run in time polynomial in the size of the hierarchy, we can get a much more reasonable response time. If the hierarchy has $g(p)$ classes, where g is a polynomial in the number of primitive predicates, p, the response time for an atomic query will be $O(g(p) * \log m)$, where m is the number of facts told the KB. This is particularly good if $g(p) = O(p)$, which is not uncommon in AI KBs, especially since it is likely that $m \gg g(p)$ in any realistic KB.

These improvements are obtained at the cost of a certain loss of fidelity: there may (will) be facts that the KB is told but is unable to recall precisely. Facts

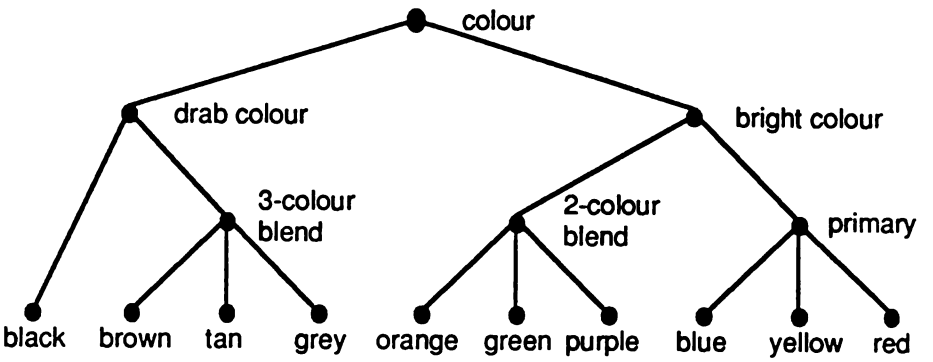


Figure 2: A hierarchy of colours

may become blurred with related facts, due to the lack of a concept precisely right for representing a given datum. Some consolation (and intuitive justification) for this loss of information comes from the fact that people exhibit similar relatedness effects [Johnson-Laird, 1982], and that information is lost proportionally to its "naturalness": very natural disjunctions will have nearby representations and little, if any, information loss; unnatural disjunctions will be stored only at the cost of significant degradation.

For example, imagine a hierarchy containing those in figures 1 and 2, in which the only common ancestor of *Colour* and *Pet* is the universal class *Thing*. In this simple pet-shop world, $Goldfish(joe) \vee Guppy(joe)$ can be perfectly represented as $Fish(joe)$, and $Gerbil(stan) \vee Hamster(stan)$ can be quite naturally approximated as $Rodent(stan)$. On the other hand, $Orange(sid) \vee Ferret(sid)$ is so bizarre that it amounts to $Thing(sid)$.

It seems reasonable to assume that, if the hierarchy is well-designed, "natural" disjunctions—those that commonly occur in commonsense reasoning—will have corresponding (or at least relatively-nearby) concepts in the hierarchy, while "unnatural" disjunctions should be only distantly connected. Since the amount of information lost in representing a disjunction is proportional to the distance between the disjuncts and their subsuming concept, this has the pleasing effect that the accuracy sacrificed to achieve the necessary efficiency for commonsense reasoning has much less impact on commonsense knowledge than on more esoteric facts.

Thus it appears to be possible to obtain significant performance improvements at the expense of sacrificing some ability to remember esoteric facts. The reason for the speed-up is simply that when told exponentially-many facts the IIKB may remember many fewer of them! The significance of the earlier complexity bounds is that such a speed-up can not be accomplished without forgetting.

3.3.2 Approximating Negation-Free Formulae

The above approximation works when the facts told the KB are uniform-argument positive clauses. There are, however, other classes of negation-free formulae that may occur in practice that can still be represented or approximated. For example, a negation-free formula involving a single constant symbol can always be converted into CNF, and then each clause can be represented separately. For example, a child choosing a pet to be named *stanley* might narrow the decision to $Brown(stan) \wedge Hamster(stan) \vee Tan(stan) \wedge Gerbil(stan)$, which can be represented in CNF as $Brown(stan) \vee Tan(stan)$, $Brown(stan) \vee Gerbil(stan)$, $Hamster(stan) \vee Tan(stan)$, and $Hamster(stan) \vee Gerbil(stan)$. Given the hierarchies of figures 1 and 2, we might remember $Drab(stan) \wedge Rodent(stan)$.

Although the final representation is brief, the process of arriving at it may still be lengthy. The results on monotone circuit complexity mentioned earlier show that conversion to CNF will exponentially lengthen some formulae (as was evident in the above example.)

Fortunately, it is possible to derive the representable information much more efficiently by avoiding the intermediate full CNF form and working incrementally. The following algorithm and theorem show how the hierarchy can contribute to efficiency at this stage.

Given a formula in disjunctive normal form (DNF), conversion to CNF proceeds by repeatedly distributing the elements of each original conjunct over the disjuncts being built up. However, as each new disjunct is added in the construction of a conjunct, the resulting partial disjunction is collapsed to its lowest super-node(s) in the hierarchy. Whenever such a collapsed disjunction corresponds to a super-node of a conjunct

already constructed, it is abandoned. If there are only polynomially many predicates in the hierarchy (as a function of the number of primitive ones), then the results always collapse sufficiently to yield a polynomial-time conversion algorithm.[6]

To illustrate the algorithm, reconsider the formula $Brown(stan) \wedge Hamster(stan) \vee Tan(stan) \wedge Gerbil(stan)$, which has the CNF expansion given above. This would be converted as follows (we elide the argument, *stan*): $Brown \vee Tan$ becomes *Drab*; $Brown \vee Gerbil$ meet at *Thing*, which is discarded; similarly for $Hamster \vee Tan$; while $Hamster \vee Gerbil$ becomes *Rodent*. Notice that the resulting atoms, *Rodent* and *Drab* coincide with the result of *TELL*ing the system the full CNF. This is not accidental, as the following theorem shows.

Theorem

Both the approximate and correct (explosive) conversions to CNF, followed by *TELL*ing the resultant clauses, result in exactly the same KB, assuming the clauses in the CNF are uniform. ■

3.3.3 Approximating Questions and Answers

The previous two sections showed how facts told to the KB can be approximated, thus significantly decreasing the size of the KB stored, and thereby improving the complexity of query processing. Approximation can also be applied to the query itself, and in the query processing phase to yield additional improvements.

It is also possible to gain speed by returning answers that only approximate what the KB knows. Both "upper-bound" (unsound but complete) and "lower-bound" (incomplete but sound) answers can be given, allowing the system to bracket the correct answer and estimate its error, as suggested in [Lipski, 1979, Imielinski, 1987]. Upper-bound answers to clausal queries are obtained by querying the predicates corresponding to the lowest nodes that subsume the query, which can be done in $O(n * \log m)$ time. For example, using the hierarchy of Figure 1, the query $Fish\text{-}eater(Fred) \vee Rodent\text{-}eater(Fred)$ can be approximated by the query $Carnivore(Fred)$. This may suffice for many situations, even though it may be answered **Yes** if Fred is a dog. Since the HKB is aware that an approximation is being presented, answers can be appropriately conditioned (e.g., "Well, he *is* a carnivore.").

Similarly lower-bound answers to disjunctive (not necessarily clausal) queries can be obtained by querying each disjunct in turn, resulting in an $O(k*n*\log m)$ time query. Such answers amount to ignoring ways that an affirmative answer might be *implicit* in the knowledge base. For example, if told $Green(sid)$ and $Reptile(sid)$, a lower bound answer to $Green(sid) \wedge Snake(sid) \vee Green(sid) \wedge Turtle(sid)$ would be obtained by looking at the individual possibilities. The result would be **Unknown** (**Yes** ∧ **Unknown** ∨ **Yes** ∧ **Unknown**), even though the correct answer could be obtained by querying $Reptile(sid) \wedge Green(sid)$.

Such approximate answers are frequently valuable. For example, in determining whether a prospective pet poses any threat to the menagerie, it may suffice simply to check whether it is a carnivore, rather than checking *Rodent-Eater*, ...—despite the fact that this will unjustly besmirch the dogs' reputation. Conversely, to see if you can satisfy a request for a goldfish and a turtle, or a guppy and a snake, or a guppy and a turtle, or a snake and a goldfish, it might be enough to check each possibility—possibly resulting in the answer "unknown"—rather than figuring out that the customer would actually be happy with any fish and any reptile!

Earlier, we discussed a strategy for answering non-clausal queries by expanding them to full clausal form, possibly causing an exponential increase in the size of the query, and then answering each clause separately. We can, however, apply the approximate conversion to clausal form to query-answering.

Theorem

Approximate conversion used with *ASK* returns the same "upper-bound" answers returned by conversion to CNF and querying the predicates subsuming the individual clauses. ■

To summarize: by using the approximation, we lose the ability to distinguish between certain given facts, and between certain questions. However, we gain desired speed, and well-motivated forgetting and errors.

This approach amounts to the decision that the HKB subsystem will avoid potentially expensive deductions, even at the cost of giving up its ability to always be right. Recall, however, that the general architecture into which the subsystem we are proposing is envisioned as fitting is one of a problem-solver that uses the HKB for efficient retrieval and reasoning. It is expected that the problem-solver will have other means at its disposal for dealing with "unnatural" and other non-uniform disjunctions. The HKB is able to detect when it is given data outside its area of expertise; this can be used to condition the problem-solver's confidence in answers the HKB may produce should it be pushed to work in those areas despite its limitations.

4 Single-Predicate Uniform Clauses

The previous section dealt with the approximation of uniform clauses of the form $Dog(a) \vee ...Cat(a)$. Here, we wish to consider a "dual" of these sentences: those

[6] It is straightforward to extend the algorithm to non-DNF ground formulae; all that is needed is to recursively apply it to DNF-subformulae, starting from the inside out.

where a single predicate is disjoined over different individuals (e.g., $Unhealthy(salt) \vee Unhealthy(pepper)$, $Bought(alex, aToyota) \vee Bought(alex, aChrysler)$, $Bought(alex, aHonda) \vee Bought(david, aHonda), \ldots.$) Such sentences appear to be especially useful in describing simple uncertainties or alternative participants in activities and plans.

4.1 The General Theory

As before, there are complexity results that provide lower bounds on how efficiently *any* KB could answer queries, when told facts of the form $P(a) \vee \ldots \vee P(z)$,[7] if we insist on correct query processing.

Theorem

Given a theory consisting of single-predicate uniform clauses

- if the language of queries is similarly restricted, then queries can be processed in time linear in the total size of the query plus the theory. (Again, query processing is not affected by assuming the GCWA.)
- determining if the answer is "yes" to single-predicate queries of the form $P(a_1) \wedge P(b_1) \vee P(a_2) \wedge P(b_2) \vee \ldots$ is co-NP-hard in both the size of the query and the size of the KB. ■

Thus, the situation is at least as bad as in the case of uniform-argument clauses.

4.2 Encoding Clauses in the Hierarchy

Once again we can encode the problem of reasoning with such clausal theories as querying certain HKBs. The construction of the appropriate hierarchy is based on the observation that a clause of the form $P(a) \vee P(b)$ can be converted to the dual form $\langle a \rangle(\overline{P}) \vee \langle b \rangle(\overline{P})$. In this dual form, we have primitive predicates of the form $\langle c \rangle$ (corresponding to original constants, c), and constants of the form $\overline{Q}$ (corresponding to original unary predicates, Q). The intended interpretation is for $\langle c \rangle(\overline{Q})$ to hold in the dual theory iff $Q(c)$ holds in the original domain. Consider now providing a *dual hierarchy* corresponding to the complete lattice of all the subsets of original constants; $\langle a, b \rangle(x)$ would be interpreted as $\langle a \rangle(x) \vee \langle b \rangle(x)$. We can use this dual hierarchy to obtain an exact encoding of the information in uniform-predicate clauses. For example, $Unhealthy(salt) \vee Unhealthy(pepper)$ is represented as $\langle salt, pepper \rangle(\overline{Unhealthy})$. We can similarly obtain dual forms of those quantifier-free queries that are uniform in single predicates.

It is not hard to convert the algorithms in Section 2 to answer clausal queries for this modified representation. The process once again is sound and complete for clausal queries with complexity polynomial in the number of type classes.

Existential queries present special problems. For example, $\exists x.\ P(x)$ would, unfortunately, become $\exists \langle x \rangle.\ \langle x \rangle(\overline{P})$, which is not first-order. Such queries must, therefore, first be converted into quantifier-free formulae, using the assumption that the theory and domain are finite.

4.3 Approximation Using HKBs

We are again in the position of using HKBs as theorem provers for a certain class of formulae. Again, we can increase efficiency (though lose accuracy) by significantly reducing the size of the hierarchy, which can be exponential in the number of constants of the original theory. Since approximation will occur if some subsets of individuals are not represented in the dual hierarchy, which subsets should be preserved?

To help explain this, we change our notation (and at the same time avoid creating the dual hierarchy) by making $Unhealthy[\langle salt, pepper \rangle]$ synonymous with $\langle salt, pepper \rangle(\overline{Unhealthy})$. We assume that the original (non-dual) hierarchy has unary predicates describing the types of objects, as is common in AI. We can approximate dual predicates such as $\langle a, b, c \rangle$ by a class that contains them as instances. Picking the lowest such class(es) would obviously be best.

For example, $Unhealthy[\langle salt, pepper \rangle]$ (recall that this stands for $Unhealthy(salt) \vee Unhealthy(pepper)$) might be approximated as $Unhealthy[\overline{CommonSpice}]$, if the class $CommonSpice$ had salt and pepper as instances, effectively asserting that some $CommonSpice$ is $Unhealthy$. Similarly, $Bought(alex, car32) \vee Bought(alex, car54)$ would be represented by asserting $Bought[alex, \overline{AmericanCar}]$, if car32 and car54 were instances of the class $AmericanCar$. Note that the accuracy of approximations might change over time, in this case, if the extensions of the type predicates change.

Alternatively, we can consider (for the first time in this paper) the addition of new classes to the hierarchy—a not uncommon operation in AI [Brachman *et al*, 1985]. For example, we might wish to *define new classes* that characterize the set of objects, $\{a, b, c\}$, over which the disjunction occurs, in terms of their properties—a problem that has been extensively studied in Machine Learning. The creation of such classes allows the (momentarily) precise representation of the disjunction, but runs the risk, in the long term, of increasing the number of concepts in the hierarchy (and therefore decreasing efficiency).[8]

[7] In this case, the unary predicate notation also stands for predicates with all arguments but one identical.

[8] The type hierarchy may not provide sufficiently-restrictive classes for some useful disjunctions. We are investigating a secondary approximation mechanism that may alleviate this problem without undue complexity, related to Imielinski's [1987] ideas of restricted unification.

As before, we may lose exact information about the individuals involved in the assertion, and retain generalizations. These generalizations, as the example above illustrates, are quite reasonable. Once again, we gain in efficiency: for example, queries of the form $P(a_1) \wedge P(b_1) \vee P(a_2) \wedge P(b_2) \vee ...$, which previously required exponentially-many operations in the number of uniform clauses asserted, can be answered by i) approximating that query in clausal form, and ii) answering the resulting query. Each step takes polynomial time in the length of the query and number of classes in the hierarchy.

5 Summary and Related Work

Semantic networks and terminologic logics are well-known examples of hierarchically organized KBs in Artificial Intelligence [Brachman *et al*, 1985]. The chief novelty of our use of HKBs is the proper treatment of the GCWA, and consideration of non-atomic updates and queries.

The approximation process, whereby some assertions are inexactly represented to increase efficiency of reasoning, has been studied by Imielinski [1986, 1987] and Plaisted [1981], among others. In particular, Imielinski [1987] creates equivalence classes of individuals (something like grouping them into classes), and then considers an abstracted KB where these equivalence classes are the new individuals. This resembles our treatment of single-predicate uniform clauses, except that we do not insist that classes be mutually exclusive, and in fact allow a hierarchy of classes in order to obtain the best approximation. Our approach also differs from Imielinski's by efficiently treating a significantly broader class of queries, and by examining a continuum of approximations from the trivial (but requiring only a simple database lookup) to the perfect (but with no performance improvement), with an attendant notion of graceful degradation of performance.

Imielinski also provides a framework for studying the error introduced by such approximations—the set of queries answered differently with and without approximation. We plan to investigate this error in our case, but note that it is possible for the HKB to bound the error for any particular query, based on the way the query interfaces with the hierarchy.

Concerning unique-argument uniform clauses, our approximation can be viewed as a case of telling information to a KB that has a different vocabulary—the higher-level classes of the hierarchy. This general view of approximation has also been considered by Plaisted and by Imielinski. To our knowledge, however, they have not considered the problem of mapping general queries into this abstracted framework—our approximation to L_{ASK} = general positive ground clauses—nor exploited special relationships between the predicates of the internal language (note that the original predicates remain part of the internal language in our case!).

We see the chief contribution of our work as the combination of two ideas: i) The provision of algorithms that are provably correct and complete when the hierarchy is a complete subset lattice (which means it is exponentially large), and ii) The property of the algorithms that pruning this hierarchy (more or less severely), improves performance (less or more dramatically), but induces a (lesser or greater) degradation in the accuracy of the KB—providing a wide spectrum of possible approximations and performance. Although the representation language is rich enough to admit unacceptable worst-case complexity, we have outlined techniques that retain acceptable performance at the cost of some degradation in the soundness and/or completeness of the representation and/or query answering.

Unlike some other approaches to achieving tractable reasoning with worst-case intractable formalisms (c.f. [Frisch, 1988]), our approach has the advantage that its departures from sound and complete reasoning are motivated by relationships in the knowledge base. The structure of the system's knowledge plays a significant role in determining the system's performance in assimilating new knowledge and in reasoning about what it knows. In particular, the more closely-related a concept available to hang new facts on, or to start retrieval from, the better the system's performance.

Just as significantly, approximation is not imposed arbitrarily, but according to the natural class hierarchy of application domains. Such hierarchies have been a staple component of almost every knowledge representation scheme in AI and Cognitive Science, and are independently motivated and necessary.

6 Open Problems and Future Work

As mentioned earlier, the results described here are part of a larger investigation of efficient knowledge representation. In that context, there are many open questions related to integrating HKBs into a more general reasoning system. These are largely common to any hybrid reasoning system involving incomplete components, and we do not go into them here. There are, however, also several open questions about HKBs, which we touch on below.

Negation: We have provided no mechanism for telling explicit negative facts to the HKB. While we believe that the negation provided by the GCWA will be sufficient for many commonsense problems, we are exploring adding some facility for explicit negation. Since adding full negation can dramatically increase the complexity of reasoning, however, we must be careful to avoid falling off a "computational cliff." It may be possible, however, to extend L_{TELL} to include conjunctions of negative, ground, atomic facts without sacrificing efficiency. Such conjunctions would further specify existing knowledge, provided the hierarchy

supplied a concept corresponding to the refinement.

Similarly, there appears to be no problem with allowing atomic negative queries. It is not clear how to deal with more complex negative queries, however, since the GCWA, which we model in our query-answering, is not complete with respect to its minimal model semantics for more complex negative queries [Shepherdson, 1988].

Non-Uniform Clauses: We have shown how to handle shared-predicate and shared-argument uniform clauses in HKBs. There is an obvious question of whether these techniques can be combined to represent clauses of the form $Pa \vee Qb$. While we see no reason why this should not be possible, we have not yet worked out the details. Nor have we convinced ourselves that the intuitive motivation for this work, based on "natural" disjunctions, applies equally to disjunctions of this form.

Assumptions About Hierarchies: The assumption of the exhaustiveness of the hierarchy (according to which each non-primitive class is defined to be equivalent to the disjunction of its subclasses) figures prominently in our representation and reasoning mechanisms. Technically, it is a relatively simple trick to add an additional "dummy" subclass (which is not visible to the user) to any class that we do not wish to consider exhausted by its children.

The GCWA induces another assumption about the hierarchy: that classes that share no common sublcasses/instances are distinct. This assumption cannot be totally discarded without abandoning the GCWA, although it might be possible to relax it for particular classes. Since we have no compelling examples where it is desirable to avoid assuming the disjointness of two classes without forming a concept for the class consisting of their intersection, we have not explored this possibility.

Updates: We have not begun to address the obviously important question of how changes to the concept hierarchy interact with the approximated knowledge of the HKB. This is particularly important in the case of single-predicate uniform clauses, since learning of new members of a type may weaken previous approximations.

Acknowledgements

This work arose from a joint research project with R. Brachman, H. Kautz, and H. Levesque, investigating tractable reasoning. We are indebted to them for many fruitful discussions and fecund concepts. We are also indebted to Ravi Boppana for pointing us to results on circuit complexity.

References

[Agrawal *et al*, 1989] Agrawal, R., Borgida, A., and Jagadish, H. V., "A transitive closure compression technique", to appear in *Proc 1989 SIGMOD Conference*, Portland, OR.

[Brachman *et al*, 1985] Brachman, R.J., Fikes, R.E., and Levesque, H.J., "KRYPTON: A Functional Approach to Knowledge Representation", in Brachman and Levesque (eds): *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, 1985, 411–429.

[Brachman & Levesque, 1984] Brachman, R.J. and Levesque, H.J., "A Fundamental Tradeoff in Knowledge Representation and Reasoning", in Brachman & Levesque (eds): *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, 1985, 41–70.

[Clark, 1978] Clark, K., "Negation as Failure" in Gallaire, H. and Minker, J. (eds), *Logic and Data Bases*, Plenum Press, 1978, 293–322.

[Codd, 1979] Codd, E.F., "Extending the Database Relational Model to Capture More Meaning", *ACM Trans. on Database Systems 4(4)*, December 1979, 397–434.

[Etherington *et al*, 1989] Etherington, D.W., Borgida, A., Brachman, R.J, and Kautz, H., "Vivid Knowledge and Tractable Reasoning", submitted for publication.

[Frisch, 1988] Frisch, A.M., *Knowledge Retrieval as Specialized Inference*, Technical Report UIUCDCS-R-88-1404, Department of Computer Science, University of Illinois at Urbana-Champaign, February 1988.

[Imielinski, 1986] Imielinski, T., "Abstraction in query processing", Rutgers University, Department Computer Science TR.178, March 1986.

[Imielinski, 1987] Imielinski, T., "Domain abstraction and limited reasoning", IJCAI-87, 997–1003.

[Imielinski, 1988] Imielinski, T., "Incomplete Deductive Databases", submitted for publication, 1988.

[Johnson-Laird, 1982] Johnson-Laird, P.N., "Ninth Bartlett Memorial Lecture. Thinking as a Skill", *Quarterly J. of Experimental Psychology 34A*, 1–29.

[Levesque, 1984] Levesque, H.J., "Foundations of A Functional Approach to Knowledge Representation", *Artificial Intelligence 23*, 1984, 155–212.

[Levesque, 1986] Levesque, H.J., "Making Believers out of Computers", *Artificial Intelligence 30(1)*, October 1986, 81–108. (Originally given as the "Computers and Thought" lecture at IJCAI-85.)

[Lipski, 1979] Lipski, W., Jr., "On Semantic Issues Connected with Incomplete Information Databases", *ACM Transactions on Database Systems 4*, 262–296, 1979.

[Minker, 1982] Minker, J., "On Indefinite Databases and the Closed-World Assumption", *Proc. Sixth Conf. on Automated Deduction*, New York, 7–9 June 1982, Springer-Verlag, NY.

[Nilsson, 1983] Nilsson, N.J., "Artificial Intelligence Prepares for 2001", *The AI Magazine 4*, Winter 1983, 7–14.

[Pippinger, 1978] Pippinger, N., "The complexity of monotone Boolean functions", *Math. Systems Theory 11*, 289–316.

[Pippinger & Fischer, 1979] Pippinger, N. and Fischer (1979), "Relations among complexity measures," *J. ACM 26*, 361–381.

[Plaisted, 1981] Plaisted, D., "Theorem proving with abstraction", *Artificial Intelligence 16*, 1981, 47–108.

[Reiter, 1984] Reiter, R., "Towards a Logical Reconstruction of Relational Database Theory", in M.L. Brodie, J. Mylopoulous, and J.W. Schmidt (eds): *On Conceptual Modelling*, Springer-Verlag, NY, 191–233.

[Selman & Kautz, 1988] Selman, B., and Kautz, H.A., "The Complexity of Model-Preference Default Theories", *Proc. Canadian Soc. for Computational Studies of Intelligence-88*, Edmonton, Alberta, June 6–10, 1988, 102–109.

[Shepherdson, 1988] Shepherdson, J. C., "Negation in Logic Programming", in J. Minker (ed), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, Los Altos, CA, 19–88.

Some Results Concerning the Computational Complexity of Abduction

Tom Bylander, Dean Allemang, Michael C. Tanner, and John R. Josephson*
Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio

Abstract

The problem of abduction is to find the best explanation of a set of data or observations. In this paper we focus on one type of abduction in which the best explanation is the most plausible conjunction of hypotheses that explains all the data. We then present several computational complexity results demonstrating that very restrictive conditions must be satisfied for this type of abduction to be tractable (solvable in polynomial time). Determining the plausibility and explanatory coverage of hypotheses must be tractable, there cannot be many incompatibility relationships or cancellation effects between individual hypotheses, and plausibility comparison between composite hypotheses must be logically weak.

1 Introduction

The problem of abduction is to find the best explanation of a set of data or observations. Diagnosis has been characterized as an abduction problem [Charniak and McDermott, 1985, de Kleer and Williams, 1987, Josephson et al., 1987, Pearl, 1987, Pople, 1973, Reggia et al., 1983, Reiter, 1987]. For example, in medical diagnosis, the signs and symptoms of the patient and additional data gathered during the diagnostic process should be explained by the final diagnosis.

In a previous paper [Allemang et al., 1987], we demonstrated that certain classes of abduction problems are computationally tractable, while other classes of abduction problems are intractable (NP-hard). Below we review these results, clarify the assumptions underlying them, and present new results which concern canceling symptoms and plausibility comparison. These results are relevant to all the diagnosis algorithms cited above.

Our methodology is to formally characterize abduction as a problem of finding the most plausible composite hypothesis (a conjunction of individual hypotheses) that explains all the data. Then we consider several classes of problems of this type, the classes being differentiated by the way hypotheses interact. We demonstrate that the time complexity of each class is polynomial (tractable) or NP-hard (intractable). Outlines of proofs are in the appendix as well as the NP-complete formulations of the problems.

Our results show that this type of abduction faces several obstacles. Incompatibility relationships or cancellation interactions between hypotheses makes abduction intractable. Our previous result pointing out a tractable class of abduction problems assumes a weak heuristic for comparing the plausibility of composite hypotheses. This assumption is crucial because normative plausibility comparison also makes abduction intractable.

We are driven to two conclusions. (1) Forget about a general algorithm for optimal abduction; perhaps a more naturalistic or satisficing conceptualization of the abduction problem should be adopted instead. (2) Hope that certain domain-dependent characteristics permit tractable abduction; in particular, abduction is tractable if composite hypotheses are guaranteed to be small or if strong domain knowledge can rule out most individual hypotheses.

2 Notation, Definitions, and Assumptions

Before reviewing our results it will be useful to have the following notational conventions and definitions. These definitions substantially extend and refine those in our previous paper.

d_x will stand for a datum, e.g., a symptom. D_x will stand for a set of data. h_x will stand for a individual hypothesis, e.g., a hypothesized disease. H_x will stand for a composite hypothesis, e.g., an hypothesized set of diseases.

*This work has been supported by the National Library of Medicine under grant LM-04298, the National Science Foundation through a graduate fellowship, and the Defense Advanced Research Projects Agency, RADC contract F30602-85-C-0010.

An *abduction problem* is a tuple $\langle D_o, H_{all}, e, pl \rangle$, where D_o is the observed data to be explained, H_{all} is the set of all individual hypotheses, e is a map from subsets of H_{all} to subsets of D_o ($e(H)$ is the data that H, if true, explains), and pl is a map from subsets of H_{all} to a set of plausibility values. That is, $e(H)$ measures the *explanatory coverage* of H, and $pl(H)$ measures the *plausibility* of H. For the purpose of this definition and the results below, it does not matter whether $pl(H)$ is a probability, a measure of belief, a fuzzy value, a degree of fit, or a symbolic likelihood. The only requirements are that the values of pl are partially ordered and that $H_i \subseteq H_j$ implies $pl(H_i) \geq pl(H_j)$.

H is *complete* if $e(H) = D_o$. That is, H explains all the data to be accounted for.

H is *parsimonious* if $\nexists H_i \subset H (e(H) \subseteq e(H_i))$. That is, no proper subset of H explains the same data as H does.

H is *a best explanation* if:

$$\begin{aligned} &\text{complete}(H) \wedge \text{parsimonious}(H) \wedge \\ &\nexists H_i \subseteq H_{all} \, (\text{complete}(H_i) \wedge \text{parsimonious}(H_i) \wedge \\ &\qquad pl(H_i) > pl(H)) \end{aligned}$$

In other words, no other complete and parsimonious composite hypothesis has a higher plausibility than H. It is just "*a* best" because pl might not impose a total ordering over composite hypotheses (e.g., because of probability intervals or qualitative likelihoods). Consequently, several composite hypotheses might satisfy these conditions. Most abduction algorithms attempt to satisfy this property (given suitable assumptions about e and pl). Some algorithms search for *all* complete and parsimonious composite hypotheses [de Kleer and Williams, 1987, Reiter, 1987]. This definition formalizes the notion of best explanation in Josephson et al. [1987].

We assume the existence and tractability of the following processes:

- a process that can perform the function e;
- a process that can perform the "inverse" of e for a datum, i.e., determine what individual hypotheses can contribute to the explanation of a datum (for convenience, we refer to this function as e^{-1}) ; and
- a process that can perform the function pl for individual hypotheses (but see Pearl [1987] concerning the tractability of this process).

Clearly, the tractability of these processes is central to abduction, since it is difficult to find plausible hypotheses explaining the data if it is difficult to compute e, e^{-1}, and pl.

The key factors, then, that we consider in the complexity of finding a best explanation are properties of abduction problems that allow or prevent tractable computation given that e, e^{-1}, and pl can be computed "easily." That is, given a particular class of abduction problems, how much of the space of composite hypotheses must be searched to find a best explanation? As demonstrated below, intractability is the usual result in classes of problems that involve significant interaction between individual hypotheses affecting the results of e and pl for composite hypotheses.

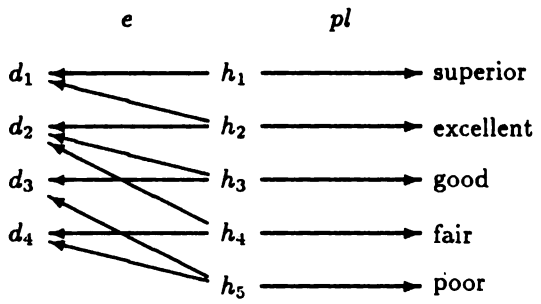


Figure 1: Example of an Abduction Problem

We should note that these definitions and assumptions oversimplify several aspects of abduction. For example, we define composite hypotheses as conjunctions of individual hypotheses. In reality, the relationships between the parts of an abductive answer can be much more complex, both logically and causally. Another example is that nothing is presumed about how e and pl are computed or how they are related. Despite these simplifications, we believe that our analysis provides powerful insights concerning the computational complexity of abduction.

We shall use the following example to facilitate our discussion:

$$\begin{aligned} H_{all} &= \{h_1, h_2, h_3, h_4, h_5\} \\ D_o &= \{d_1, d_2, d_3, d_4\} \\ e(h_1) &= \{d_1\} \qquad pl(h_1) = \text{superior} \\ e(h_2) &= \{d_1, d_2\} \qquad pl(h_2) = \text{excellent} \\ e(h_3) &= \{d_2, d_3\} \qquad pl(h_3) = \text{good} \\ e(h_4) &= \{d_2, d_4\} \qquad pl(h_4) = \text{fair} \\ e(h_5) &= \{d_3, d_4\} \qquad pl(h_5) = \text{poor} \end{aligned}$$

Figure 1 is a pictorial representation of the example. The values of pl should simply be interpreted as indicating relative order of plausibility. Assuming no interactions between hypotheses, $\{h_2, h_3, h_5\}$ would be a complete explanation, but not parsimonious since h_3 is superfluous. $\{h_2, h_3\}$ would be a parsimonious explanation, but not complete since it doesn't explain d_4. $\{h_1, h_3, h_4\}$ or $\{h_2, h_5\}$ could be considered best explanations, depending on how plausibilities combine.

Using these definitions and assumptions, we first discuss how properties of e affect the tractability of finding best explanations, and then consider properties of pl.

3 Complexity of Completeness and Parsimony

We consider several different classes of problems and state the complexity of finding a complete explanation and finding a parsimonious subset of a composite hypothesis.

3.1 Independent Problems

The simplest problems assume that an individual hypothesis explains a specific set of data regardless of what other individual hypotheses are being considered. INTERNIST-I [Miller et al., 1982] and Reggia's set covering algorithm [Reggia et al., 1983] are two systems that make this assumption. The use of conflict sets [de Kleer and Williams, 1987, Reiter, 1987] also appears to make this assumption.[1]

Formally, an abduction problem is *independent* if:

$$\forall H \subseteq H_{all}\,(e(H) = \bigcup_{h \in H} e(h))$$

That is, a composite hypothesis explains a datum if and only if some component explains the datum. Assuming independence, the complete and parsimonious explanations in our example (refer to Figure 1) are $\{h_1, h_3, h_4\}$, $\{h_1, h_3, h_5\}$, $\{h_1, h_4, h_5\}$, $\{h_2, h_3, h_4\}$, and $\{h_2, h_5\}$.

In Allemang et al. [1987], we showed that: *For the class of independent problems, finding a composite hypothesis that is complete is tractable.*[2] The algorithm that we presented in that paper performs this function using $O(d(h + d\log d))$ steps (any call to e or e^{-1} is counted as one step), where d is the size of D_o and h is the size of H_{all}.

Also, we showed that: *For the class of independent problems, finding a parsimonious subset of a composite hypothesis is tractable.* In particular, once a complete composite hypothesis has been found, finding a subset which is parsimonious is tractable. Our algorithm performs this function in $O(hd\log d)$ steps.

3.2 Monotonic Problems

A more general kind of problem is when composite hypotheses can explain additional data that are not explained by any of its components. This sort of interaction can occur if two individual hypotheses have an additive interaction. For example, each of the two hypotheses can explain a small value of some measurement, but together can explain a larger measurement.

Formally, an abduction problem is *monotonic* if:

$$\forall H_i, H_j \subseteq H_{all}\,(e(H_i) \cup e(H_j) \subseteq e(H_i \cup H_j))$$

That is, a composite hypothesis does not "lose" any data explained by any of its proper subsets and might explain additional data. An independent abduction problem is monotonic, but a monotonic abduction problem is not necessarily independent. If, in our example, $\{h_2, h_3\}$ also explained d_4, then $\{h_2, h_3\}$ would also be a complete and parsimonious explanation and $\{h_2, h_3, h_4\}$ would not be.

Our previous paper showed that our algorithms for independent problems also work for monotonic problems with no change in complexity. Thus: *For the class of monotonic problems, finding a complete and parsimonious composite hypothesis is tractable.*

3.3 Incompatibility Problems

Implicit in the formal model so far is the assumption that any collection of individual hypotheses is possible. However, most domains have restrictions that invalidate this assumption. For example, an electrical switch cannot simultaneously be closed and open.

Formally, an *incompatibility abduction problem* is a tuple $\langle D_o, H_{all}, e, pl, I\rangle$, where D_o, H_{all}, e, and pl are the same as before and I is a set of two-element subsets of H_{all}, indicating pairs of hypotheses that are incompatible with each other.[3] For an incompatibility problem:

$$\forall H \subseteq H_{all}\,(\exists P \in I\,(P \subseteq H) \rightarrow e(H) = \emptyset)$$

By this formal trick, a composite hypothesis containing incompatible hypotheses explains nothing, preventing such a composite from being complete (except for trivial cases) or a best explanation. We describe an incompatibility problem as independent if composite hypotheses with no incompatibilities satisfy the independence formula. If $I = \{\{h_1, h_2\}, \{h_2, h_3\}, \{h_3, h_4\}, \{h_4, h_5\}\}$ in our example, then only $\{h_1, h_3, h_5\}$ and $\{h_2, h_5\}$ would be complete and parsimonious.

Our previous paper showed: *For the class of independent incompatibility problems, finding a complete composite hypothesis with no incompatibilities is NP-hard.* However, the same parsimony algorithm for independent and monotonic problems can be applied to incompatibility problems. Once a composite hypothesis without any incompatibilities is found, no difficulties with incompatibilities will arise by removing individual hypotheses. Thus, *For the class of independent incompatibility problems, finding a parsimonious*

[1] Each conflict set corresponds to a datum to be explained, and the elements of the conflict set are the hypotheses that can independently explain the datum.

[2] To be precise, we should say that it is tractable to determine whether a complete composite hypothesis exists and that, if one does exist, finding it is also tractable.

[3] Incompatible pairs are the most natural case, e.g., one hypothesis of the pair is the negation of the other. Incompatible triplets (any two of the three, but not all three) and so on are conceivable, but allowing these possibilities in the formal definition do not affect the complexity results.

Table 1: Computational Complexity of Obtaining Completeness and Parsimony

class of problems	condition to achieve		
	completeness	parsimony	best explanation
independent	P	P	?
monotonic	P	P	?
incompatibility	NP	P	NP
cancellation	NP	NP	NP

P = known polynomial algorithm NP = NP-hard

subset of a composite hypothesis is tractable. Nevertheless, because of the complexity of obtaining completeness, it follows that: *For the class of independent incompatibility problems, finding a best explanation is NP-hard.*

3.4 Cancellation Problems

To be an independent or monotonic abduction problem, one hypotheses cannot "cancel" a datum that another hypothesis would otherwise explain. However, cancellation can occur when one hypothesis can have a subtractive effect on another. This is common in medicine, e.g., in the domain of acid-base disorders one disease might explain an increased blood pH, and another might explain a decreased pH, but together the result might be a normal pH [Patil et al., 1982]. Some of our new results concern problems of this class.

Formally, we define a *cancellation* abduction problem as a tuple $\langle D_o, H_{all}, e, pl, f, g\rangle$. f is a map from H_{all} to subsets of D_o indicating what data each hypothesis "explains." g is another map from H_{all} to subsets of D_o indicating what data each hypothesis "cancels." $d \in e(H)$ iff the number of hypotheses in H that explain d outnumber the hypotheses that cancel d. That is:

$$d \in e(H) \leftrightarrow |\{h: h \in H \wedge d \in f(h)\}| > |\{h: h \in H \wedge d \in g(h)\}|$$

In our example, if we let $f = e$ for individual hypotheses and if $g(h_1) = \{d_3\}$, $g(h_2) = \{d_4\}$, and $g(h_3) = g(h_4) = g(h_5) = \emptyset$, then the only complete and parsimonious explanations would be $\{h_1, h_3, h_5\}$ and $\{h_2, h_4, h_5\}$.

Admittedly, this is a oversimplified model of cancellation effects. Nevertheless: *For the class of cancellation problems, finding a complete composite hypothesis is NP-hard.* Also, it turns out that: *For the class of cancellation problems, finding a parsimonious subset of a composite hypothesis is NP-hard.* Clearly then: *For the class of cancellation problems, finding a best explanation is NP-hard.*

Table 1 summarizes the results of this section.

4 Complexity of Plausibility

To apply the definition of best explanation, the plausibilities of composite hypotheses need to be comparable. We consider three plausibility criteria based on comparing the plausibilities of the components of the composite hypotheses. Our new results are the formalization of the criteria and all but one of the complexity results. Our previous paper did not consider two of the criteria and implicitly assumed the appropriateness of the other one (the one called best-for-some).

The first criterion is a qualitative version of normative plausibility comparison; the other two criteria are heuristic. None of the criteria are sensitive to whether composite hypotheses are causally coherent. Nevertheless, they help to delimit what classes of abduction problems are tractable.

4.1 The Best-small Plausibility Criterion

Everything else being equal, smaller composite hypotheses are preferable to larger ones, and more plausible individual hypotheses are preferable to less plausible ones. Thus, in the absence of other information, it seems reasonable to compare the plausibility of composite hypotheses based on their sizes and the relative plausibilities of their components. When a conflict occurs, e.g., one composite hypothesis is smaller, but has less plausible components, no ordering can be imposed.

Formally, the *best-small* plausibility criterion is defined as[4]:

$$pl(H) > pl(H') \leftrightarrow \exists m: H \rightarrow H' \,(m \text{ is 1-1} \wedge \forall h \in H\,(pl(h) \geq pl(m(h))) \wedge (|H| = |H'| \rightarrow \exists h \in H\,(pl(h) > pl(m(h)))))$$

To be more plausible according to best-small, the components of H need to be matched to the components

[4]This and the following definitions of plausibility criteria implicitly assume that abduction problems are independent and that plausibilities of individual hypotheses are totally ordered. More general definitions have been constructed, but are much less understandable.

of H' so that the components of H are at least as plausible as their matches in H'. If H and H' are the same size, then in addition some component in H must be more plausible than its match in H'. In our example, $\{h_1, h_3, h_4\}$ and $\{h_2, h_5\}$ would be the best explanations (assuming that the problem is independent).

We have demonstrated that: *For the class of independent problems using the best-small criterion, finding a best explanation is NP-hard.*

It is possible to reduce from the best-small criterion to any normative probability theory in which more plausible or fewer individual hypotheses can increase probability, which of course includes Bayesian, Dempster-Shafer, and fuzzy logic theory. Hence: *For the class of independent problems using probability theory, finding a best explanation is NP-hard.* That is, an algorithm for finding a best explanation for independent problems using probability theory can be applied to finding a best explanation for independent problems using the best-small criterion. For example, if we let the probabilities of individual hypotheses be independent from each other, then a best explanation for the probability problem would be a best explanation for the best-small problem. Thus, if the best-small problem is intractable (NP-hard), then so is the probability problem. Note that this is true even if there are assumptions that make calculating probabilities tractable.

4.2 The Best-for-all Plausibility Criterion

Because normative comparison (i.e., the best-small criterion) is intractable, it is appropriate to consider heuristic criteria. One of the characteristics of the best-small criterion is that it does not depend on which individual hypotheses explain which data. A heuristic alternative is to compare composite hypotheses based on how well each datum is explained. If one composite hypothesis has a more plausible explanation of a datum than some other composite hypothesis, we shall say that the datum *discriminates* in favor of the first composite hypothesis. Before defining the second plausibility criterion based on this idea, we first define when an individual hypothesis contributes to explaining a datum in the context of a composite hypothesis and when a datum discriminates between composite hypotheses.

h contributes to explaining *d* within H (contributes(h, d, H)) iff:

$$h \in H \wedge d \in e(h) \wedge \forall h_i \in H\ (d \in e(h_i) \rightarrow pl(h) \geq pl(h_i))$$

That is, h is the most plausible hypothesis in H that explains *d*.

d discriminates in favor of H over H' (discriminates(d, H, H')) if:

$$\exists h \in H, h' \in H'\ (\text{contributes}(h, d, H) \wedge \text{contributes}(h', d, H') \wedge pl(h) > pl(h'))$$

That is, H's explanation of *d* is more plausible than H''s explanation of *d*.

The second plausibility criterion, *best-for-all*, prefers composite hypotheses in which every datum is explained by as plausible an individual hypothesis as possible. Formally:

$$pl(H) > pl(H') \leftrightarrow \exists d \in D_o\ (\text{discriminates}(d, H, H')) \wedge \nexists d \in D_o\ (\text{discriminates}(d, H', H))$$

That is, some datum discriminates in favor of H, and no datum discriminates in favor of H'. In our example, $\{h_1, h_3, h_4\}$ and $\{h_2, h_3, h_4\}$ would be best explanations using best-for-all. As the example demonstrates, best explanations using the best-small criterion are not a subset of those using the best-for-all criterion (and vice versa).

Even though this is a heuristic, we have shown that: *For the class of independent problems using the best-for-all criterion, finding a best explanation is NP-hard.*

4.3 The Best-for-some Plausibility Criterion

The intractability of the best-for-all criterion is partly due to the fact that any datum and any individual hypothesis that contributes to that datum can be used for making a discrimination. In contrast, the third plausibility criterion requires that a discrimination be based on all the data that an individual hypothesis contributes to. Before defining the criterion, we first need a predicate that defines this version of discrimination.

D set-discriminates in favor of H over H' (set-discriminates(D, H, H')) if:

$$D \neq \emptyset \wedge \exists h' \in H'\ (\forall d \in D_o\ (d \in D \leftrightarrow \text{contributes}(h', d, H')) \wedge \forall d \in D\ (\exists h \in H\ (\text{contributes}(h, d, H) \wedge pl(h) > pl(h'))))$$

Informally, H has a more plausible explanation for every datum explained by some h' in H'.

The third plausibility criterion, *best-for-some*, prefers composite hypotheses in which every component is the most plausible for some datum. Formally:

$$pl(H) > pl(H') \leftrightarrow \exists D \subseteq D_o\ (\text{set-discriminates}(D, H, H')) \wedge \nexists D \subseteq D_o\ (\text{set-discriminates}(D, H', H))$$

Informally, every datum explained by some h' in H' is more plausibly explained by H, and every h in H

Table 2: Computational Complexity of Finding a Best Explanation

property of e	property of pl		
	best-for-some	best-for-all	best-small
independent	P	NP	NP
monotonic	P	NP	NP
incompatibility	NP	NP	NP
cancellation	NP	NP	NP

P = known polynomial algorithm NP = NP-hard

explains some datum as plausibly as H' does. In our example, $\{h_1, h_3, h_4\}$ and $\{h_2, h_3, h_4\}$ would be again be best explanations. Best-for-some best explanations are not a subset of best-for-all best explanations, though. If instead $pl(h_2)$ = good, then $\{h_2, h_3, h_4\}$ would still be a best explanation using best-for-some because h_2 explains d_2 as plausibly as any other hypothesis, but not using best-for-all because d_2 no longer discriminates in its favor. Best-for-all is not a subset of best-for-some, either. If in our example $e(h_3) = \{d_1, d_2, d_3\}$, then $\{h_3, h_4\}$ would be the only best explanation using best-for-some, but $\{h_1, h_4, h_5\}$ and $\{h_2, h_5\}$ would also be best explanations using best-for-all.

Our previous paper [Allemang et al., 1987], implicitly assuming this criterion, showed that: *For the classes of independent and monotonic abduction problems using the best-for-some plausibility criterion, finding a best explanation is tractable.* The algorithm presented in Allemang et al. [1987] performed this problem in $O(hm(h + m \log m))$

Table 2 summarizes our results.

4.4 Enumeration of Best Explanations

It is interesting to note that finding a best explanation using best-small or best-for-all is tractable if the plausibilities of individual hypotheses are all different from each other and if the plausibilities are totally ordered. Thus, part of the complexity comes from having to choose between different individual hypotheses that have similar plausibilities and whose explanations overlap. This still creates a problem for probability theories using real numbers because errors in probability values will make small differences insignificant.

There is an additional problem to consider even if a best explanation can be found—that of finding alternative best explanations or determining that no more exist. Finding alternatives is usually an important part of abduction because of the need to know whether there is only one good explanation, or if there are several, the need to perform tests that will discriminate between the alternatives.

Unfortunately, even assuming that the plausibilities of individual hypotheses are all different and totally ordered, generating alternative best explanations is intractable in the following cases:

- *For the class of monotonic problems, given a set of complete and parsimonious explanations, it is NP-hard to find an additional complete and parsimonious explanation.*
- *For the class of independent problems using the best-small plausibility criterion, given a set of best explanations, it is NP-hard to find an additional best explanation.*
- *For the class of independent problems using the best-for-all plausibility criterion, given a set of best explanations, it is NP-hard to find an additional best explanation.*

We suspect that it is intractable to enumerate best-for-some best explanations, but we have not yet been able to demonstrate it.

5 Discussion

These results seem rather depressing for abductive reasoning. We believe that few domains (i.e., classes of problems) satisfy the independent or monotonic property. Probability theory appears to guarantee intractability for abduction. However, there are several mitigating factors.

One factor is that incompatibility relationships and cancellation effects might be sufficiently sparse so that it is not expensive to search for complete and parsimonious composite hypotheses. However, plausibility requirements might still result in intractability.

Another factor might be that in practice composite hypotheses are usually small. However, our results suggest that considerable search will in general be required. If h is the number of individual hypotheses and l is the limit on the size of composite hypotheses, then $O(h^l)$ composite hypotheses are possible, which becomes formidable if h is large and $l > 1$.

A third factor is that in a specific domain, there might be sufficient knowledge to rule out a large number of individual hypotheses or, more generally, to assign high plausibility values to the "right" individual hypotheses. For example, if rule-out knowledge can reduce the number of individual hypotheses from h to $\log h$, then the problem is tractable.[5] It is important to note that this factor does not simply call for "more knowledge," but knowledge of the right type, in this case, *rule-out* knowledge. Additional knowledge *per se* does not reduce complexity. For example, more knowledge about incompatibilities or cancellations makes abduction harder.

One expert system that takes advantage of many of these factors is RED, which performs red-blood cell antibody identification [Smith et al., 1985, Allemang et al., 1987]. RED's problem solving only attempts to satisfy best-for-some, general classes of hypotheses are ruled out whenever possible, plausibility values are assigned to hypotheses by evaluating the evidence efficiently, and the search for a best explanation takes this information into account as well as information about which data are more important to explain [Josephson et al., 1987]. INTERNIST-I [Miller et al., 1982] has many of these same properties, except that it does not perform any explicit rule-out and assumes that problems are independent. Virtually all knowledge-based systems that do diagnosis perform evidence evaluation, but few perform an efficient search for composite hypotheses that are complete, parsimonious, and "best." We are not saying that RED has a tractable algorithm for an intractable class of problems, but that RED takes advantage of many factors that may allow for tractable problem solving in specific domains.

If there are no tractable algorithms for a class of abduction problems, then there is no choice but do abduction heuristically (unless one is willing to wait for a long time). One must accept the best explanation that can be found with the computational resources available. We believe this will lead to the adoption of a more naturalistic or satisficing conceptualization of abduction, such as that of Josephson and Goel [1988]. Unfortunately, the result is that some data might not be explained, incompatibilities might exist, and there might be more plausible and parsimonious explanations. Perhaps one mark of intelligence is being able to act despite the lack of optimal solutions.

Our results show that abduction, characterized as finding the most plausible composite hypothesis that explains all the data, is generally an intractable problem. Thus, it is futile to hope for a general tractable algorithm that produces optimal answers for diagnosis and other abduction problems. To solve an abduction problem efficiently, the problem must have certain factors that make the problem tractable, and the problem solving strategy must take advantage of those factors. Elucidating such factors and strategies cannot be wholly a search for the right epistemology, but must take computational complexity into account.

[5] This is not the same as the "rule-out" performed in the diagnosis algorithms of de Kleer & Williams [1987] and Reiter [1987]. Determining that an individual hypothesis is insufficient to explain all the observations, i.e., that it's not a member of every conflict set, does not prevent the individual hypothesis from being in composite hypotheses.

A Outlines of Proofs

In this appendix, we list the complexity results discussed in the paper and outline a proof for each of them. We assume that the functions e, e^{-1}, and pl are tractable (see Section 2). The reader is forewarned that many of the reduction proofs do not provide much insight on the intuitive reasons underlying the complexity results. The proofs are only intended to be mathematically correct, not intuitively helpful.

Theorem 1 *For the class of independent problems, it is tractable to determine whether a complete composite hypothesis exists and to construct it if one does exist.*

Simply check if $e(H_{all}) = D_o$. If so, H_{all} is a complete composite hypothesis. If not, no complete composite hypothesis exists.

Theorem 2 *For the class of independent problems, finding a parsimonious subset of a composite hypothesis is tractable.*

The following algorithm solves this problem in $O(hd)$ steps where d is the size of D_o and h is the size of H_{all} and each call to e, e^{-1}, and pl is counted as one step. Testing for set equality can be performed in linear time using bit vectors.

H is the composite hypothesis to be parsimonized.
W stands for the working composite hypothesis.

$W \leftarrow H$
For each $h \in H$
    If $e(W \backslash \{h\}) = e(W)$ then
        $W \leftarrow W \backslash \{h\}$
Return W

Suppose that the result of this algorithm was a non-parsimonious composite hypothesis H. Then, because the problem is independent, there is some $h \in H$ such that $e(H \backslash \{h\}) = e(H)$. However, the algorithm would have removed h from H (or any superset of H) in just this case, which is a contradiction. Thus, for the class of independent problems, the algorithm produces parsimonious hypotheses.

Theorem 3 *For the class of monotonic problems, it is tractable to determine whether a complete composite*

hypothesis exists, to construct it if one does exists, and to find a parsimonious subset.

The algorithms for the previous two theorems also apply to this theorem. The crucial reason underlying the tractability of monotonic problems is that $H \subseteq H'$ implies $e(H) \subseteq e(H')$. Thus, in monotonic problems, no composite hypothesis can explain more than H_{all}, and parsimony can be performed by iterating over individual hypotheses. Without monotonicity, such guarantees cannot be made.

Theorem 4 *For the class of independent incompatibility problems, it is NP-complete to determine whether there is a complete composite hypothesis.*

We prove the NP-completeness of this problem by reducing from 3SAT [Garey and Johnson, 1979]: given a statement in propositional calculus in disjunctive normal form, in which each term has at most three factors, find an assignment of variables that makes the statement true.

Let S be a statement in propositional calculus in 3SAT form. Let $\{u_1, u_2, \ldots, u_m\}$ be the variables used in S. Let n be the number of terms in S. An equivalent independent incompatibility problem $IP = \langle D_o, H_{all}, e, pl, I\rangle$ can be constructed by:

$$\begin{aligned} D_o &= \{d_1, d_2, \ldots, d_n\} \\ H_{all} &= \{h_1, h_1', h_2, h_2', \ldots, h_m, h_m'\} \\ e(h_i) &= \{d_j\text{: } u_i \text{ is a factor in the } j^{th} \text{ term}\} \\ e(h_i') &= \{d_j\text{: } \neg u_i \text{ is a factor in the } j^{th} \text{ term}\} \\ I &= \{\{h_1, h_1'\}, \{h_2, h_2'\}, \ldots, \{h_m, h_m'\}\} \end{aligned}$$

If H is a complete explanation for IP, then S is satisfied by the following assignment.

$$u_i = \begin{cases} \text{true} & \text{if } h_i \in H \\ \text{false} & \text{otherwise} \end{cases}$$

Below, we reduce satisfaction of completeness in incompatibility problems to a number of problems. For convenience, we shall assume that the incompatibility problems have the same form as IP: the problem is independent except for incompatibilities, each $h \in H_{all}$ is an element of some $P \in I$, and the incompatibilities are disjoint, i.e., the intersection of any two elements of I is $\emptyset$. We refer to this kind of problem as 3SAT incompatibility problems.

Theorem 5 *For the class of independent incompatibility problems, finding a parsimonious subset of a composite hypothesis is tractable.*

If a composite hypotheses contains no incompatible pairs, then the problem can be reduced to obtaining parsimony for an independent problem, which is tractable by Theorem 2. If a composite hypotheses contains an incompatible pair, then by definition it explains nothing and $\emptyset$ is the parsimonious subset ($\emptyset$ is the smallest composite hypotheses that explains nothing).

Theorem 6 *For the class of cancellation problems, it is NP-complete to determine whether there is a complete composite hypothesis.*

This can be shown by reduction from 3SAT incompatibility problems. Let $IP = \langle D_o, H_{all}, e, pl, I\rangle$ be a 3SAT incompatibility problem. An equivalent cancellation problem $CP = \langle D_o', H_{all}', e', pl', f', g'\rangle$ can be constructed by:

$$\begin{aligned} D_o' &= D_o \cup \{d_P\text{: } P \in I\} \\ H_{all}' &= H_{all} \cup \{h', h''\} \\ f'(h) &= \begin{cases} e(h) & \text{if } h \in H_{all} \\ \{d_P\text{: } P \in I\} & \text{if } h \in \{h', h''\} \end{cases} \\ g'(h) &= \begin{cases} \{d_P\text{: } \exists P \in I(h \in P)\} & \text{if } h \in H_{all} \\ \emptyset & \text{if } h \in \{h', h''\} \end{cases} \end{aligned}$$

Cancellation interactions are created so that each incompatible pair in IP effectively become incompatible in CP. Thus, CP has a complete explanation iff IP has a complete explanation. In particular, if H is a complete explanation for CP, then $H \cap H_{all}$ is a complete explanation for IP.

Theorem 7 *For the class of cancellation problems, it is NP-complete to determine whether there is a more parsimonious subset of a composite hypothesis.*

This can be shown by reduction from 3SAT incompatibility problems. Let $IP = \langle D_o, H_{all}, e, pl, I\rangle$ be a 3SAT incompatibility problem, and let $CP = \langle D_o', H_{all}', e', pl', f', g'\rangle$ be a cancellation problem constructed from IP as follows:

$$\begin{aligned} H_{all}' &= H_{all} \cup \{h', h'', h^*, h^{**}\} \\ D_o' &= D_o \cup \{d_P\text{: } P \in I\} \\ &\quad \cup \{d_h\text{: } h \in H_{all}' \setminus \{h^*, h^{**}\}\} \\ f'(h) &= \begin{cases} e(h) \cup d_h & \text{if } h \in H_{all} \\ \{d_P\text{: } P \in I\} & \text{if } h \in \{h', h''\} \\ D_o \cup \{d_P\text{: } P \in I\} & \text{if } h = h^* \\ \{d_h\text{: } h \in H_{all}' \setminus \{h^*, h^{**}\}\} & \\ & \text{if } h = h^{**} \end{cases} \\ g'(h) &= \begin{cases} \{d_P\text{: } \exists P \in I(h \in P)\} & \text{if } h \in H_{all} \\ \{d_h\text{: } h \in H_{all}' \setminus \{h^*, h^{**}\}\} & \text{if } h = h^* \\ \emptyset & \text{if } h \in \{h', h'', h^{**}\} \end{cases} \end{aligned}$$

This construction is similar to the previous one except that additional data and hypotheses are included so that H_{all}' is a complete explanation. However, any

other complete explanation (which must be more parsimonious than H'_{all}) cannot include h^- and must satisfy cancellation interactions equivalent to IP's incompatibilities. In particular, if H is a complete explanation for CP, then $H \cap H_{all}$ is a complete explanation for IP.

Theorem 8 *For the class of independent problems using the best-small plausibility criterion, given a complete and parsimonious explanation, it is NP-complete to determine whether a better complete and parsimonious explanation exists.*

This can be shown by reduction from 3SAT incompatibility problems. Let $IP = \langle D_o, H_{all}, e, pl, I \rangle$ be a 3SAT incompatibility problem, and let $IP' = \langle D'_o, H'_{all}, e', pl' \rangle$ be an independent problem constructed from IP as follows:

Let f_1 be a function from I to H_{all}, such that $\forall P \in I(f_1(P) \in P)$, i.e., f_1 chooses one hypothesis from each pair in I. Let H_1 be the set of hypotheses that f_1 chooses, i.e, $H_1 = \cup_{P \in I}\{f_1(P)\}$. Now define IP' as:

$$
\begin{aligned}
D'_o &= D_o \cup \{d_P\colon P \in I\} \\
H'_{all} &= H_{all} \cup \{h'\} \\
e'(h) &= \begin{cases} e(h) \cup \{d_P\colon h \in P\} & \text{if } h \in H_{all} \\ D_o \backslash e(H_1) & \text{if } h = h' \end{cases} \\
pl'(h) &= \text{good} \quad \forall h \in H'_{all}
\end{aligned}
$$

Now $H_1 \cup \{h'\}$ is a complete and parsimonious explanation of size $n+1$, where $n = |I|$. Because all the hypotheses have the same plausibility, any better explanation using the best-small plausibility criterion must be of size n or less. However, one hypothesis out of each pair in I (a total of n hypotheses) is needed to explain $\{d_P\colon P \in I\}$. Hence, to get a better complete and parsimonious explanation, h' must be excluded and only one hypothesis out of each pair in I can be accepted. Thus, any better complete and parsimonious explanation for IP' using best-small must also be a complete explanation for IP.

Theorem 9 *Finding a best explanation for the class of independent problems using probability theory is NP-hard.*

The best explanation for this problem would be the most probable composite hypothesis explaining all the data. This can be reduced from the best-small problem by assigning probabilities consistent with the plausibility ratings and letting the probabilities of the individual hypotheses be conditionally independent from each other. The best explanation for this probability problem is also a best explanation under the best-small plausibility criterion.

Theorem 10 *For the class of independent problems using the best-for-all criterion, given a complete and parsimonious explanation, it is NP-complete to determine whether a better complete and parsimonious explanation exists.*

This can be reduced from 3SAT incompatibility problems. Let $IP = \langle D_o, H_{all}, e, pl, I \rangle$ be a 3SAT incompatibility problem, and let $IP' = \langle D'_o, H'_{all}, e', pl' \rangle$ be an independent problem constructed from IP as follows:

Again let f_1 be a function from I to H_{all}, such that f_1 chooses one hypothesis from each pair in I. Let f_2 be a function from I to H_{all}, such that f_2 chooses other hypothesis from each pair in I. Now define IP' as:

$$
\begin{aligned}
H'_{all} &= H_{all} \cup \{h_P\colon P \in I\} \cup \{h'\} \\
D'_o &= D_o \cup \{d'_P, d''_P\colon P \in I\} \\
e'(h) &= \begin{cases} e(h) \cup \{d'_P\colon h = f_1(P)\} \\ \quad \cup \{d''_P\colon h = f_2(P)\} & \text{if } h \in h_{all} \\ \{d'_P, d''_P\} & \text{if } h = h_P \\ D_o & \text{if } h = h' \end{cases} \\
pl'(h) &= \begin{cases} \text{fair} & \text{if } h = h' \\ \text{good} & \text{if } h \in H_{all} \\ \text{excellent} & \text{if } \{h_P\colon P \in I\} \end{cases}
\end{aligned}
$$

Now $H = \{h_P\colon P \in I\} \cup \{h'\}$ is a complete and parsimonious explanation. Any better complete and parsimonious explanation using the best-for-all plausibility criterion must include $\{h_P\colon P \in I\}$, otherwise some d'_P and d''_P would discriminate in H's favor. Clearly then, h' must be excluded. Also, at most one element out of each $P \in I$ can be part of a better explanation because any P makes the corresponding h_P superfluous. Thus, if a better complete and parsimonious explanation exists, then IP has a complete explanation. In particular if H' is a better complete and parsimonious explanation using best-for-all than H for IP', then $H' \backslash \{h_P\colon P \in I\}$ is a complete explanation for IP.

Theorem 11 *For the class of independent problems using the best-for-some criterion, finding a best explanation is tractable.*

Theorem 12 *For the class of monotonic problems using the best-for-some criterion, finding a best explanation is tractable.*

The algorithm below adapted from Allemang et al. [Allemang et al., 1987] is tractable and obtains a best explanation for the class of monotonic problems using the best-for-some plausibility criterion. Since the class of independent problems is a subset of the class of monotonic problem, this algorithm also applies to independent problems. It performs in $O(h(h+d))$ steps where d is the size of D_o and h is the size of H_{all} and each call to e, e^{-1}, and pl is counted as one step.

W stands for the working composite hypothesis.
D stands for the data yet to be explained.
Nil is returned if no best explanation exists.

$W \leftarrow \emptyset$
$D \leftarrow D_o$

Use most plausible hypotheses to construct a complete explanation.

Until $D = \emptyset$
  Let d be an element of D
  $C \leftarrow e^{-1}(d) \backslash W$
  If $C = \emptyset$ then
    Return nil
  else $W \leftarrow W \cup$ most plausible element of C
  $D \leftarrow D \backslash e(W)$

Find a parsimonious subset of W.

For each $h \in W$ from least to most plausible
  If $e(W \backslash \{h\}) = e(W)$ then
    $W \leftarrow W \backslash \{h\}$
Return W

This algorithm also finds a best explanation for the class of *ordered* monotonic problems using the best-small plausibility criterion, as well as the class of *ordered* monotonic problems using the best-for-all plausibility criterion. Thus, for the purpose of generating a best explanation, ordered monotonic problems using either the best-small or best-for-all plausibility criteria can be treated as a monotonic best-for-some problem.

Theorem 13 *For the class of monotonic problems, given a set of complete and parsimonious explanations, it is NP-complete to determine whether an additional complete and parsimonious explanation exists.*

This can be reduced from 3SAT incompatibility problems. Let $IP = \langle D_o, H_{all}, e, pl, I \rangle$ be a 3SAT incompatibility problem, and let $MP = \langle D'_o, H'_{all}, e', pl' \rangle$ be a monotonic problem constructed from IP as follows.

$$\begin{aligned} D'_o &= D_o \\ H'_{all} &= H_{all} \\ e'(H) &= \begin{cases} D_o & \text{if } \exists P \in I (P \subseteq H) \\ e(H) & \text{otherwise} \end{cases} \\ pl'(h) &= pl(h) \end{aligned}$$

It turns out that I is a set of complete and parsimonious explanations for MP. Consequently, any other complete and parsimonious explanation can only have at most one hypothesis from each pair $P \in I$. Thus, it would also be a complete and parsimonious explanation for IP.

Theorem 14 *For the class of ordered independent problems using the best-small plausibility criterion, given a set of best explanations, it is NP-complete to determine whether an additional best explanation exists.*

This reduction is very similar to that of Theorem 8. Let $IP = \langle D_o, H_{all}, e, pl, I \rangle$ be a 3SAT incompatibility problem, and let $IP' = \langle D'_o, H'_{all}, e', pl' \rangle$ be an independent problem constructed from IP as follows:

Let f_1 be a function from I to H_{all}, such that f_1 chooses one hypothesis from each pair in I. Let H_1 be the set of hypotheses that f_1 chooses. Let f_2 be a function from I to H_{all} that chooses the other hypothesis from each pair in I. Now define IP' as:

$$\begin{aligned} D'_o &= D_o \cup \{d_P : P \in I\} \\ H'_{all} &= H_{all} \cup \{h'\} \\ e'(h) &= \begin{cases} e(h) \cup \{d_P : h \in P\} & \text{if } h \in H_{all} \\ D_o \backslash e(H_1) & \text{if } h = h' \end{cases} \\ &\forall h \in H_{all} (pl'(h') > pl'(h)) \\ &\forall P \in I (pl'(f_1(P)) > pl'(f_2(P))) \end{aligned}$$

The remaining orderings for pl' do not matter. Now $H = H_1 \cup \{h'\}$ is a best explanation of size $n + 1$, where $n = |I|$. Because one hypothesis out of each pair in I (a total of n hypotheses) is needed to explain $\{d_P : P \in I\}$, and because H includes the more plausible hypothesis of each pair and the most plausible hypothesis overall, no other complete and parsimonious explanation of size $n + 1$ or greater can be as good as H. Hence, to construct another best explanation, h' must be excluded and only one hypothesis out of each pair in I can be accepted. Thus, another best explanation for IP' exists if and only if IP has a complete explanation.

Theorem 15 *For the class of ordered independent problems using the best-for-all plausibility criterion, given a set of best explanations, it is NP-complete to determine whether an additional best explanation exists.*

This can be reduced from 3SAT incompatibility problems. Let $IP = \langle D_o, H_{all}, e, pl, I \rangle$ be a 3SAT incompatibility problem, and let $IP' = \langle D'_o, H'_{all}, e', pl' \rangle$ be an independent problem constructed from IP as follows:

Again let f_1 be a function from I to H_{all}, such that f_1 chooses one hypothesis from each pair in I. Let f_2 be a function from I to H_{all}, such that f_2 chooses the other hypothesis from each pair in I. Now define IP' as:

$$\begin{aligned} H'_{all} &= H_{all} \cup \{h'_P, h''_P : P \in I\} \\ D'_o &= D_o \cup \{d'_P, d''_P : P \in I\} \\ e'(h) &= \begin{cases} e(h) \cup \{d'_P : h = f_1(P)\} \\ \quad \cup \{d''_P : h = f_2(P)\} & \text{if } h \in h_{all} \\ \{d'_P, d''_P\} & \text{if } h = h'_P \\ D_o \cup \{d'_P, d''_P\} & \text{if } h = h''_P \end{cases} \end{aligned}$$

$\forall h \in H_{all}, h' \in \{h'_P, h''_P: P \in I\}(pl'(h') > pl'(h))$
$\forall h' \in \{h'_P: P \in I\}, h'' \in \{h''_P: P \in I\}(pl'(h') > pl'(h''))$

The remaining orderings for pl' do not matter. Now for each $P \in I$, $\{h''_P\} \cup \{h'_{P'}: P \neq P'\}$ is a best explanation using the best-for-all plausibility criterion. Let $\mathcal{H}$ be this set of best explanations. Any additional best explanation must include $\{h_P: P \in I\}$, otherwise some explanation in $\mathcal{H}$ would be better. Consequently, every h''_P must be excluded. Also, at most one element out of each $P \in I$ can be part of another best explanation because any P makes the corresponding h'_P superfluous. Thus, if another best explanation exists, then IP has a complete explanation. In particular if H' is another best explanation using best-for-all for IP', then $H' \backslash \{h'_P: P \in I\}$ is a complete explanation for IP.

References

[Allemang et al., 1987] Allemang, D., Tanner, M. C., Bylander, T., and Josephson, J. R. (1987). On the computational complexity of hypothesis assembly. In *Proc. Tenth Int. Joint Conf. on Artificial Intelligence*, pages 1112–1117, Milan.

[Charniak and McDermott, 1985] Charniak, E. and McDermott, D. (1985). *Introduction to Artificial Intelligence*. Addison-Wesley, Reading, Mass.

[de Kleer and Williams, 1987] de Kleer, J. and Williams, B. C. (1987). Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability*. W. H. Freeman, New York.

[Josephson et al., 1987] Josephson, J. R., Chandrasekaran, B., Smith, J. W., and Tanner, M. C. (1987). A mechanism for forming composite explanatory hypothesis. *IEEE Trans. Systems, Man, and Cybernetics*, 17(3):445–454.

[Josephson and Goel, 1988] Josephson, J. R. and Goel, A. (1988). Tractable abduction. Technical report, Lab. for AI Research, CIS Dept., Ohio State Univ., Columbus, OH.

[Miller et al., 1982] Miller, R. A., Pople, J. E., and Myers, J. D. (1982). INTERNIST-I, An experimental computer-based diagnostic consultant for general internal medicine. *New England J. of Medicine*, 307:468–476.

[Patil et al., 1982] Patil, R. S., Szolovits, P., and Schwartz, W. B. (1982). Modeling knowledge of the patient in acid-base and electrolyte disorders. In Szolovits, P., editor, *Artificial Intelligence in Medicine*, pages 191–226. Westview Press, Boulder, Colorado.

[Pearl, 1987] Pearl, J. (1987). Distributed revision of composite beliefs. *Artificial Intelligence*, 33(2):173–215.

[Pople, 1973] Pople, H. E. (1973). On the mechanization of abductive logic. In *Proc. Third Int. Joint Conf. on Artificial Intelligence*, pages 147–152, Stanford.

[Reggia et al., 1983] Reggia, J. A., Nau, D. S., and Wang, P. (1983). Diagnostic expert systems based on a set covering model. *Int. J. Man-Machine Studies*, 19:437–460.

[Reiter, 1987] Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95.

[Smith et al., 1985] Smith, J. W., Svirbely, J. R., Evans, C. A., Strohm, P., Josephson, J. R., and Tanner, M. C. (1985). RED: A red-cell antibody identification expert module. *J. of Medical Systems*, 9(3):121–138.

On the Appearance of Sortal Literals: a Non Substitutional Framework for Hybrid Reasoning

A G Cohn
Dept Computer Science
University of Warwick
COVENTRY, CV4 7AL
UK
agc@cs.warwick.ac.uk

Abstract

Many Sorted Logics have been widely recognized as useful in Artificial Intelligence for computational as well as pragmatic reasons. There are many different formulations of many sorted logic; most do not allow characteristic (i.e. sortal) literals. This paper is largely devoted to a discussion of characteristic literals in many sorted logic: whether they should appear (and why) and in what form. The final section is devoted to a consideration of the implications of allowing characteristic literals in the broader context of many sorted logic as a hybrid reasoning system.

1. Introduction

In a *many sorted* logic the universe of discourse is divided up into subsets, and variables are restricted to range over these subsets (called sorts) rather than the whole universe as is normally the case. Declarations regarding the sortal behaviour of all the function symbols (and possibly the predicate symbols as well) must be made; this allows the notion of a *well sorted formula* to be defined (intuitively: one in which the sort of every term matches the sort of the argument position in which it occurs). Many sorted logics are an important kind of logic for Artificial Intelligence. There are many reasons for this; the main one perhaps is the computational efficiency that can be achieved when deduction takes place in a many sorted formulation compared to an unsorted one – for example see Cohn [1985], Walther [1985], Ohlbach and Schmidt-Schauss [1985] or Frisch [1986]. Another important reason often given for using many sorted logics is their notational convenience, e.g. see McDermott [1982] or Hayes [1985]. A further advantage of many sorted logics is the fact that one can perform (certain kinds of) integrity checking of assertions and queries to a knowledge base efficiently – e.g. see Minker and McSkimin[1977] or Reiter [1981]. Many sorted logics vary in their expressive power, where by expressive power we mean the amount of knowledge that can be represented by special purpose declarations rather than by ordinary logical formulae. Cohn [1989] discusses the main dimensions along which many sorted logics differ:

- The structure of the set of sorts.
- The way in which the sortal behaviour of the non logical symbols is described.
- The way in which sortal constraints on variables are specified.
- Whether the sort of a term is constrained to be a subsort of the sort of the argument position it occurs in.
- Whether characteristic literals are allowed.

It is the last of these with which this paper is concerned. Traditionally, many sorted logics do not allow characteristic literals (i.e. literals whose predicate symbol is a sort symbol) to appear in formulae. However, greater expressiveness results when such literals are allowed as we shall see. This paper is devoted to an exploration of characteristic literals in many sorted logic. The title of the paper is deliberately ambiguous and can be interpreted in the following ways:

1) Are characteristic literals allowed to appear in a formula?
2) Under what circumstances are characteristic literals required? (We shall see that sometimes characteristic literals may appear dynamically.)
3) What syntactic form do characteristic literals take? (We shall discuss several different options).

We shall discuss each of these questions in more detail in the three sections following the introduction to many sorted logic in section 2. In the final section we will consider the implications of allowing characteristic literals in the broader context of many sorted logic as a particular

This work has been supported by the SERC under grant GR/E/00273.

hybrid reasoning system.

2. Many Sorted Logic

As outlined above, a many sorted logic differs from an unsorted logic in that the universe of discourse is divided into non empty subsets, each of which is denoted by a *sort* symbol. Traditionally, the set of sorts are pairwise disjoint (or rather, their interpretations are), as in Enderton [1972]. However sorts may overlap and a sort hierarchy (ordered by set inclusion) may be formed.[1] One speaks of a sort τ_1 being a *subsort* of τ_2 if $\tau_1=\tau_2$ or it is lower in the hierarchy; in such cases we write $\tau_1 \sqsubseteq \tau_2$. It is common to insist that there is a unique most general sort, called $\top$, (read *top*) which is always interpreted as the universe of discourse. The special sort $\bot$ (read *bottom*) which is below all other sorts in the hierarchy always denotes the empty set and a term of this sort is *ill-sorted*. We write $\tau_1 \sqcup \tau_2$ to indicate the least upper bound of two sorts τ_1 and τ_2 in the hierarchy and $\tau_1 \sqcap \tau_2$ to indicate their greatest lower bound. $\tau_1 \setminus \tau_2$ indicates the relative complement of τ_2 with respect to τ_1 in a sort lattice. (We assume here that these three operators are always defined and unique).

In a many sorted logic there is always a way of attaching sortal constraints to variables. Usually, this is done locally on a per variable basis, either by predeclaring that certain variable names have a particular sort restriction or by tagging occurrences of variables in clauses with sorts. However it is also possible for variables to gain their sortal restrictions implicitly from the argument positions they occur in (e.g. as in the many sorted logic Llama [Cohn 1987]).

In a many sorted logic the non logical symbols are only defined as making sense certain arguments sorts. The sort of the result of a function is also specified. We will write $\ddot{\alpha}(\tau_1, \cdots, \tau_n) = \tau_{n+1}$ to indicate a declaration of a function symbol α being of sort τ_{n+1} when its arguments are of sort $\tau_1 \ldots \tau_n$. If more than one such declaration is allowed then the function is said to be *polymorphic*. In this case the sort of a term will vary depending on the sorts of its arguments. The sortal behaviour of a predicate symbol α may specified similarly; we can write: $\ddot{\alpha}(\tau_1, \cdots, \tau_n)$ to specify that any literal whose predicate symbol is α is well sorted when its arguments are of sort $\tau_1 \ldots \tau_n$. If we regard a predicate symbol as a Boolean valued function we can give a result sort for predicate symbols as we do for function symbols. We will use a special sort lattice, **B**, whose elements are UU, TT, FF, EE; the interpretations of these sorts are fixed and are {true, false}, {true}, {false} and the empty set respectively. Thus UU and EE are the top and bottom elements of **B** respectively. We can now write $\ddot{\alpha}(\tau_1, \cdots, \tau_n) = \tau_{n+1}$ where $\tau_{n+1} \in \mathbf{B}$, to describe the sortal behaviour of a predicate symbol. We will exploit this method of sorting predicate symbols later.

Making a logic many sorted has three main effects: on the syntax, on the semantics and on the inference machinery. Formulae which are ill-sorted (because the sorts of one or more terms do not match the sorts required for the argument positions they occupy) are deemed to be syntactically ill-formed. The semantics of a many sorted logic is such that function symbols are only total with respect to their argument sorts, and can thus be partial functions on the universe of discourse; moreover only well sorted formulae can have a Tarskian truth valuation. In the absence of characteristic literals, the effect that a many sorted logic has on the inference machinery is restricted to modifying any substitution rule for variables so that only terms of appropriate sorts can be substituted for variables.

If characteristic literals are allowed into formulae then the inference mechanism of the logic is affected in other ways too. For example, to retain completeness in a resolution based system, a rule of *characteristic resolution* must be introduced so that it is possible for two sort literals to clash even if they are not of opposite sign and even if the predicate names differ (e.g. Man(c) and Woman(c) would be contradictory if Man and Woman are sorts whose greatest lower bound is $\bot$ in the sort hierarchy). It is also possible for characteristic literals to clash partially, leaving a residue literal in the resolvent. Thus in general two positive characteristic literals $\tau_1(c)$ and $\tau_2(c)$ resolve to give $\tau_3(c)$ where τ_3 is $\tau_1 \sqcap \tau_2(c)$; if τ_3 is $\bot$ then the literal is equivalent to *false* and can be deleted and one can always insist that $\tau_3 \sqsubset \tau_1$ and $\tau_3 \sqsubset \tau_2$ without losing completeness. See the description of the many sorted logic Llama [Cohn 1983, 1987] for details of such a rule including the case when one or both literals are negative. Characteristic resolution can be viewed as an instance of *Theory Resolution* [Stickel 1985].

One can also change the definition of subsumption to take note of characteristic literals (e.g. Man(c)$\vee\Phi$ subsumes Human(c)$\vee\Phi$). Formulae may also be simplified when they contain more than one characteristic literal: clauses can be normalised so that no term is predicated by more than one sort predicate providing that the sort hierarchy is closed under $\sqcup$ (e.g. Man(c)$\vee$Woman(c) can be simplified to Human(c)). It may turn out that a clause is a tautology after such simplification (iff it contains a literal of the form $\top(\alpha)$). Again, see Cohn [1983,1987]

[1] Such logics are sometimes called *order sorted* to emphasize the fact that the sort structure can be hierarchical.

for details of these rules.

3. Are characteristic literals allowed to appear at all?

First, we should point out that characteristic literals predicating a variable are never needed in a many sorted logic since the variable can be restricted directly; so the following discussion only concerns instantiated characteristic literals (except at the end of section 5.4). It is easy to argue that allowing characteristic literals to appear at all in a formula is at odds with the notion of a many sorted logic; a many sorted logic allows knowledge about sorts to be represented separately (and reasoned about more efficiently than would be possible in an unsorted logic). Since there exists a special language for representing such information why allow redundancy? Why not force sortal knowledge to appear only in special purpose declarations and not in ordinary formula? There is much to be said for such an approach to many sorted logic. Essentially it allows the interface between the sortal 'box' (i.e. the component which contains the sortal knowledge) and the main representation to be very clean and simple. Inference can proceed on formulae as normal until such time as a substitution is required; at this point the sortal box is called to ensure that the proposed substitution is consistent with the sortal knowledge given the sortal constraints on the variables involved. Frisch [1989] has characterised this approach as the *substitutional* approach. By disallowing characteristic literals the required changes to the underlying logic are kept to the minimum: no extra inference rules are required. If the interaction between the sorts and the underlying logic is restricted to checking the well sortedness of substitutions then a sortal component can easily be added to any unsorted deductive system and the meta theoretic results (such as completeness) can be easily transferred to the sorted system.

If the sorted logic cannot be accommodated within a substitutional framework then the completeness results may have to be derived anew for each deductive system to which the sort mechanism is added. (For example, see Cohn [1983] for a completeness proof for Llama, a non substitutional many sorted logic).

The main disadvantage of the substitutional approach is that expressiveness is limited because sort literals cannot appear in normal formulae. An example of a problem where this might be useful is Lewis Carroll's *Salt and Mustard Problem* [Bartley 1977] discussed below. This problem can be viewed as having to deduce what taxonomic class certain things are in. In a many sorted logic without characteristic literals one cannot reason about the sorts of terms, these are predeclared, so such reasoning has to be performed entirely by ordinary unsorted inference. However by making the taxonomic classes sorts, and allowing characteristic literals, one can take advantage of the built in sort structure. Although the Salt and Mustard Problem is an artificial one, performing taxonomic classification is an important task for knowledge based systems [Levesque and Brachman 1987].

3.1. The 'Salt and Mustard problem'

This problem was previously discussed in the the problem corner of the very first issue of the Journal of Automated Reasoning (Lusk and Overbeek 1985). It was originally formulated by Lewis Carroll [Bartley 1977]. We repeat[2] the formulation of the problem here for the reader's convenience.

> *Five friends, Barry, Cole, Dix, Lang and Mill, agreed to meet every day at a certain table-d'hote. They devised the following rules, to be observed whenever beef appeared on the table.*
>
> *(1) If Barry takes salt, then either Cole or Lang takes one only of the two condiments, salt and mustard: if he takes mustard, then either Dix takes neither condiment, or Mill takes both.*
>
> *(2) If Cole takes salt, then either Barry takes only one condiment, or Mill takes neither: if he takes mustard, then either Dix or Lang takes both.*
>
> *(3) If Dix takes salt, then either Barry takes neither condiment or Cole takes both: if he takes mustard, then either Lang or Mill takes neither.*
>
> *(4) If Lang takes salt, then either Barry or Dix takes only one condiment, if he takes mustard, then either Cole or Mill takes neither.*
>
> *(5) If Mill takes salt, then either Barry or Lang takes both condiments: if he takes mustard, then either Cole or Dix takes only one.*
>
> *The Problem is to discover whether these rules are compatible; and, if so, what arrangements are possible.*
>
> *[N.B. In this Problem, it is assumed that the phrase 'If Barry takes salt' allows of two possible cases, viz (1) 'He takes salt only'; (2) 'He takes both condiments'. And so with all similar phrases.*
>
> *It is also assumed that the phrase, 'Either Cole or Lang takes one only of the two condiments'*

[2] Two typographical errors in Lusk and Overbeek [1985] are corrected here.

allows of three possible cases, viz. (l) 'Cole takes one only, Lang takes both or neither'; (2) 'Cole takes both or neither, Lang takes one only'; (3) 'Cole takes one only, Lang takes one only'. And so with all similar phrases.

It is also assumed that every rule is to be understood as implying the words 'and vice versa'. Thus the first rule would imply the addition 'and, if either Cole or Lang takes only one condiment, then Barry takes salt'.]

The answer is:
Dix and Cole take neither salt nor mustard.
Barry takes both.
Lang takes mustard but not salt.
Mill takes salt but not mustard.

Lusk and Overbeek [1985] formulated the problem straightforwardly in clausal form thus:

*Both(x) means 'x takes both salt and mustard'.[3]
*Neither (x) means 'x takes neither salt nor mustard'.
*Oneof(x) means 'x takes exactly one of salt and mustard'.
*Salt(x) means 'x takes salt'.
*Mustard(x) means 'x takes mustard'.

*Exactly one of Both, Neither and Oneof holds;
1) Both(x) ∨ Neither(x) ∨ Oneof(x);
2) ¬Oneof(x) ∨ ¬Both(x);
3) ¬Oneof(x) ∨ ¬Neither(x);
4) ¬Both(x) ∨ ¬Neither(x);

*Definition of Oneof;
5) Oneof(x) → Salt(x) ∨ Mustard(x)
6) Oneof(x) → ¬Salt(x) ∨ ¬Mustard(x)

*Definition of Neither;
7) Neither(x) → ¬Salt(x)
8) Neither(x) → ¬Mustard(x)
9) ¬Salt(x) ∧ ¬Mustard(x) → Neither(x)

*Definition of Both;
10) Both(x) → Salt(x)
11) Both(x) → Mustard(x)
12) Salt(x) ∧ Mustard(x) → Both(x)

13) Salt(Barry) → Oneof(Cole) ∨ Oneof(Lang)
14) Mustard(Barry) → Neither(Dix) ∨ Both(Mill)
15) Oneof(Cole) → Salt(Barry)
16) Oneof(Lang) → Salt(Barry)
17) Neither(Dix) → Mustard(Barry)
18) Both(Mill) → Mustard(Barry)

19) Salt(Cole) → Oneof(Barry) ∨ Neither(Mill)
20) Mustard(Cole) → Both(Dix) ∨ Both(Lang)
21) Oneof(Barry) → Salt(Cole)
22) Neither(Mill) → Salt(Cole)
23) Both(Dix) → Mustard(Cole)
24) Both(Lang) → Mustard(Cole)

25) Salt(Dix) → Neither(Barry) ∨ Both(Cole)
26) Mustard(Dix) → Neither(Lang) ∨ Neither(Mill)
27) Neither(Barry) → Salt(Dix)
28) Both(Cole) → Salt(Dix)
29) Neither(Lang) → Mustard(Dix)
30) Neither(Mill) → Mustard(Dix)

31) Salt(Lang) → Oneof(Barry) ∨ Oneof(Dix)
32) Mustard(Lang) → Neither(Cole) ∨ Neither(Mill)
33) Oneof(Barry) → Salt(Lang)
34) Oneof(Dix) → Salt(Lang)
35) Neither(Cole) → Mustard(Lang)
36) Neither(Mill) → Mustard(Lang)

37) Salt(Mill) → Both(Barry) ∨ Both(Lang)
38) Mustard(Mill) → Oneof(Cole) ∨ Oneof(Dix)
39) Both(Barry) → Salt(Mill)
40) Both(Lang) → Salt(Mill)
41) Oneof(Cole) → Mustard(Mill)
42) Oneof(Dix) → Mustard(Mill)

Lusk and Overbeek comment that the problem is difficult: "The first unit clause is not derived until fairly late in the run, and then there is another long wait for the second one. In the course of the run, more than 32,000 clauses were generated and then subsumed." I gave the problem to the theorem prover ITP [Lusk and Overbeek 1984] with the standard settings and options and the set of support was exhausted after 426 interpreter cycles and 846 minutes of cpu time; the failure to derive a contradiction (given the complete proof strategy) shows that the rules are consistent; sufficient unit clauses concerning the individuals were derived after 854 minutes of cpu time[4] and 424 interpreter cycles; there were 130 clauses involved in the derivation of these unit clauses.

3.1.1. A Many Sorted Solution

Although at first sight the problem would appear to be eminently suitable for attacking with a many sorted logic since all the predicates in the problem are unary, and indeed the first 12 clauses can be regarded as defining a sort hierarchy, the rest of the clauses consist of disjunctions of ground literals and in most many sorted logics,

[3] Note that we use italic symbols (e.g. x) for individual variables.

[4] All timings were performed on a 2MIP SUN3.

sort (or *characteristic)* literals are not allowed and so the rest of the clauses cannot be expressed if the unary predicates are to be sorts. However the remaining 30 axioms are all disjunctions of ground literals. In each of these clauses each literal predicates a different constant symbol. Such clauses cannot be eliminated by special purpose declarations in any many sorted logic known to the author because the clause represents disjunctive information about the sort of more than one non-logical symbol.[5] However, if we are to forbid characteristic literals in formulae then the monadic predicates involved in such formulae cannot be sort symbols. In the many sorted logic Llama [Cohn 1987] characteristic literals can appear in normal formulae and so problems such as the Salt and Mustard Problem can exploit the sortedness of the logic. The search space is considerably reduced because certain inferences can be performed by the special purpose mechanisms in the many sorted logic.

Clauses 13 through 42 are identical to those in the unsorted formulation. but the first 12 clauses are not needed since the sort declarations take care of them (i.e. make them tautologous). The sort lattice for the problem is given below in Figure 1; note that the sort symbols Onlysalt and Onlymustard have been introduced in order to define Salt, Mustard and Oneof.

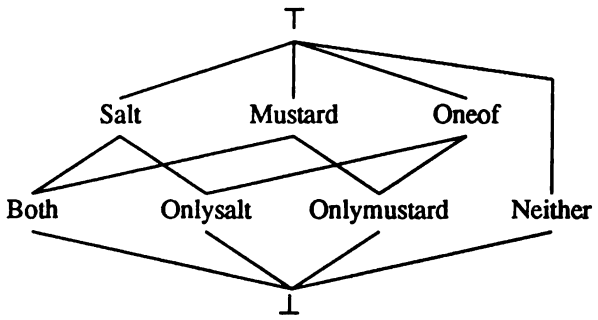


Figure 2.

Table 1 below shows the results obtained, with the figures for the unsorted logic in the ITP formulation along side; the proof strategies[6] were the same, though binary resolution was used for the Llama run and hyperesolution for the ITP run. The search space would have been yet bigger for ITP if binary resolution had been used.

	Llama	ITP
total cpu seconds	4178	51005
clauses generated	6145	>32000
interpreter cycles	237	424
number of clauses in proof	55	130

Table 1.

The statistics in the table come from a run which was automatically terminated as soon as positive unit clauses predicating a base sort of each of the individuals had been derived.

This logic puzzle, which is characteristic of any situation where one knows of a number of taxonomic classes and some rules for assigning objects to them, but not explicitly which objects are members of which classes, demonstrates the computational advantages of allowing characteristic literals in a many sorted logic. In this particular problem there were only characteristic literals in the problem formulation but in general clauses may be *mixed* i.e. contain both characteristic literals and non sortal literals. Note that Frisch [1989] actually proposes that the sort component might be represented explicitly as a set of clauses where all predicates are sort symbols. In this case one could solve the *Salt and Mustard* problem entirely within the sortal box, but this would be identical to the unsorted case, since ordinary resolution is used in his sortal box.[7]

[5] The information content of a sort declaration for a non logical symbol α can usually be expressed by a formula of the form:

$\forall(x_1 \cdots xn)\,\alpha(x_1, \cdots, xn) \rightarrow \tau_1(x_1) \wedge \cdots \wedge \tau_n(xn)$

if α is a predicate symbol whose argument sorts are τ_1 ... τ_n and

$\forall(x_1 \cdots xn)\,\tau_1(x_1) \wedge \cdots \wedge \tau_n(xn) \rightarrow \tau_{n+1}(\alpha(x_1, \cdots, xn))$

if α is a rank n function symbol which is of sort τ_{n+1} when its arguments are of sort τ_1 ... τ_n. For further details see descriptions of *relativizing* a many sorted logic (i.e. translating it into unsorted logic) in e.g. Walther [1987], Schmidt-Schauss [1988] or Cohn [1989]. In each case notice that there is at most one non logical symbol (discounting 'sort' symbols) in these formulae.

[6] Using a different proof strategy for Llama has resulted in a run about four times faster (i.e. in about 17 minutes total cpu time). The implementation of Llama has not been optimised at all; for example there is no indexing of clauses performed at present. As always, comparative timings should be treated with due respect. The figures giving the size of the search space and the length of the proof are of course independent of the implementation.

[7] It is not clear whether Frisch's logic should actually be called a many sorted logic as there is no notion of well sortedness; since his sort theory contains only sort literals there is no way to express any information about the sorts of arguments of predicate symbols: not only does he not allow characteristic literals in ordinary axioms, he also disallows ordinary predicates in sortal axioms. The sortal box can only be used to deduce sorts of terms but has no special effect on the semantics of the logic in the same way that sort declarations actually restrict the set of possible interpretations.

In this problem, the (many sorted) axiomatisation contains no variables so the usefulness of the many sortedness of the logic had nothing to do with any restriction on unification cutting out branches of the search space early, but is entirely due to the fact that the search space is smaller because 12 clauses have been treated specially as defining a sort hierarchy. Apart from the reduction of the search space engendered simply by having fewer initial axioms this means that literals clash directly when otherwise they would not have done. This could also have been achieved in Stickel's [1985] *Theory Resolution*. For example, clauses 14 and 19 resolve (on their final literals) to give

$$\text{Mustard(Barry)} \wedge \text{Salt(Cole)} \rightarrow \text{Neither(Dix)} \vee \text{Oneof(Barry)}$$

which is automatically simplified (because the first and last literals predicate the same term) to

$$\text{Both(Barry)} \wedge \text{Salt(Cole)} \rightarrow \text{Neither(Dix)}$$

Clauses 40 and 41 can be resolved together to give

$$\text{Both(Lang)} \wedge \text{Oneof(Cole)} \rightarrow \text{Both(Mill)}$$

(after simplification) which includes a residue literal, Both(Mill).

Whenever a clause is simplified by combining characteristic literals predicating the same term (as happened above) the search space is reduced because the resulting clause is shorter: branches in the search space have been collapsed.

3.2. Other uses for characteristic literals

Another use for characteristic literals in formulae is to represent what Schmidt-Schauss [1988] calls *term declarations*. A term declaration is a way of specifying the sort of a term directly rather than indirectly, i.e. by declaring the sortal behaviour of all the constituent function symbols. This is more expressive because we can write declarations such as Even:plus(x,x) which cannot be expressed in the traditional way because of the repeated variable. If characteristic literals are allowed into the logic then of course it is easy to express the information content of arbitrary term declarations (for example in the above situation we would write Even(plus(x,x))). Of course, this is not a sort declaration in the manner of Schmidt-Schauss's logic for there, a term declaration directly affects the unification algorithm, while a unit clause consisting of a characteristic literal (such as Even(plus(x,x))) does not. Of course, it is more expressive in the sense that we can write arbitrary conditions on such 'declarations' (e.g. $\Phi(x,y) \rightarrow \text{Even}(x,y)$) which is not possible in term declarations. It must be pointed out again though, that such formulae do not affect the logic in the direct way that a term declaration does. However Schmidt-Schauss shows that unification under term declarations is in general undecidable so it is not clear whether it is worth treating this kind of information specially (although he does isolate some decidable special cases).

There is a strong case for not allowing characteristic literals if we wish to restrict formulae to Horn clauses. This is because in normal Horn clauses a literal in the body of a clause could only be complementary with one in the head of another clause. However if we were to allow characteristic literals then a characteristic literal in the body of a Horn clause might be complementary with one in the body of another clause. This would inevitably lead to loss of completeness of SLD resolution which is the normal inference rule used for Horn clauses.

For example given the sort structure in Figure 2 below, the following Horn clauses are unsatisfiable

:- S4(c1)
S2(c2) :- S6(c1)
S3(c2)

but there is no SLD refutation even if characteristic resolution is available (because the literal in the body of the first clause will not unify with the head of any other clause). However the logically equivalent formulae

:- S4(c1)
S1(c1) :- S5(c2)
S3(c2)

do have an SLD refutation.

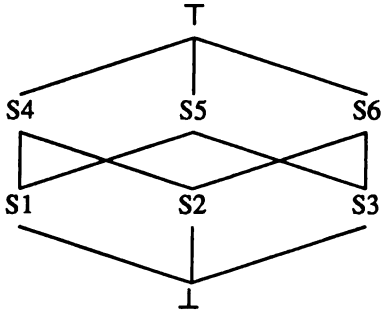


Figure 2.

3.3. Characteristic literals and the disjointness of sorts

Sometimes a problem specification does not explicitly say whether predicates which are desired to be interpreted as sorts are disjoint or not. For example in the English description of *Schubert's Steamroller* [Stickel 1986] it is not explicitly stated that foxes, wolves, caterpillars and other animals are disjoint sets. However as pointed out in Cohn [1986] in the absence of characteristic literals in the axiomatisation it can never hurt to assume that sorts not known to have a non $\perp$ glb are in fact disjoint. This is because declaring two sorts to be disjoint is equivalent to

a clause of the form $\neg\tau_1(x)\vee\neg\tau_2(x)$: i.e. one which contains only negative literals. If a many sorted axiomatisation, $\mathcal{A}$, without characteristic literals is translated to unsorted logic, only negative 'sort' literals are added to the clauses of the axiomatisation (in order to express the sortal restrictions on the variables in the clauses). Clearly clauses such as $\neg\tau_1(x)\vee\neg\tau_2(x)$ could never be used in a refutation of $\mathcal{A}$.

4. The dynamic appearance of characteristic literals

In the previous section we showed that it might be desirable to allow a many sorted logic to include formulae which contained characteristic literals because the initial problem formulation contained predicates which might be naturally regarded as sort predicates except that they occurred in formulae which could not be viewed as sort declarations for a non logical symbol. In this section we show that we might also want to allow derived formulae to contain characteristic literals (even if none of the original axioms do). Plotkin first noticed [Kowalski and Hayes 1971] that an increase in expressive power in a many sorted logic results if the sort of a term is allowed to be more general than the sort of the argument position where it occurs (this is called *overlapping* in Cohn [1987]).

For example one might define the monadic function symbol 'cr' (standing for 'closest relation') to return a Human given a Human argument. If M is a two place predicate symbol whose first argument must be a Woman and whose second a Human, then we cannot resolve $M(x,cr(y))$ with $\neg M(z,z)$ (where x, y and z are all variables) unless we allow the term $cr(y)$ (which is of sort Human) to be substituted for z which is constrained to be of sort Woman (where Woman $\sqsubset$ Human). However even if we allow this substitution, in order for the inference rule to be sound, the resolvent has to contain an additional (sort) literal (called a *prosthetic* literal in Cohn [1987]) which predicates that the term is of the more restricted sort. Thus in the example above the resolvent will contain the literal $\neg$Woman$(cr(y))$. The inference is sound because in any interpretation, either $cr(y)$ will denote a woman (in which case z has been appropriately instantiated) or $\neg$Woman$(cr(y))$ will be true. With this technique, a term α of sort τ_1 may be substituted for a variable of sort τ_2 providing $\tau_1\sqcap\tau_2\neq\bot$. Unless $\tau_1\sqsubseteq\tau_2$ then the condition that $\tau_2(\alpha)$ must be added to the instantiated clause.

Thus we see that if we relax the definition of well sortedness to allow overlapping then we need to also allow characteristic literals into formulae. Note that if overlapping is not allowed, then we are forced to reduce the granularity of the sort structure or the sort declarations for the non logical symbols. For example in the situation above we would either have to make Woman an ordinary predicate rather than a sort predicate or define M as taking a Human in the first argument position. In order to keep the same semantics for the clauses, a restriction on any variable occurring as the first argument to M is necessary: as Woman is no longer a sort, we would have to include the literal $\neg$Woman(z) in the clause containing the literal $\neg M(z,z)$. By not allowing overlapping we are forced to reduce the amount of information we can represent as special purpose sort declarations, either by not being able to say that Woman implies Human, or that $M(x,y)$ implies Woman(x).[8]

Another example of the utility of allowing overlapping arises in a Naive Physics axiomatisation of topology being developed [Randell and Cohn 1989]. One of the sorts in the theory is REGION, which denotes spatial (and/or temporal) regions. Various (binary) predicates such as C(onnects), P(art of), O(verlaps) and D(isconnected) are defined on this sort. However, when defining function symbols to represent Boolean compositions of regions one runs into the problem that the intersection of disconnected regions is not defined. If this concept is to be a function rather than a relation, then a null object has to be introduced. However there are philosophical objections to introducing the null object, especially since the theory is developed from a calculus of individuals [Clarke 1981,1985] which has no null object. The solution which can be adopted is to define a sort NULL (whose intended interpretation is the singleton subset of the universe of discourse containing the null individual) disjoint from REGION, and specify the result sort of 'prod' (the name of the intersection function) as REGION⊔NULL. Providing we allow overlapping, expressions such as

O(a,prod(a,b))

are well sorted. However we have kept the calculus clean in the sense that we have not admitted the null object as a region.

---

[8] Actually, there is a third possibility in Llama. The sortal behaviour of predicate symbols is declared by specifying the result sort is one four special Boolean sorts TT, FF, UU or EE (meaning definitely true, definitely false, not known whether true or false, and ill sorted respectively). Given this mechanism (and the ability to specify *polymorphic* predicates, i.e. give more than one declaration for any predicate) we could specify the result sort of M as UU when the arguments are both Human and FF when the first argument is a non Woman Human (i.e. a Man). However the clause involving $\neg M(z,z)$ will be tautologous (i.e. of sort TT) when z is of sort Man and therefore the variable z is automatically restricted to sort Woman and thus overlapping is still required.

Another example of the utility of allowing overlapping can be found in Cohn [1989]: a theory of lists is developed and the problem is what sort the functors head and tail should have, given the sort structure in Figure 3.

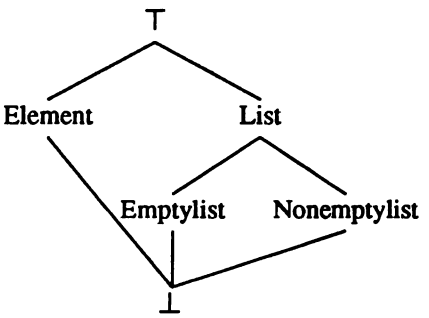


Figure 2.

Overlapping avoids the introduction of error sorts to deal with the case of taking the tail of an empty list, while specifying as much information as possible in the sortal declarations for head and tail, cons and the constant nil:

$\dot{cons}$(Element,List) = Nonemptylist
$\dot{head}$(Nonemptylist) = Element
$\dot{tail}$(Nonemptylist) = List
$\dot{nil}$() = Emptylist

All these functions have their natural definitions on the sort structure and expressions such as head(tail(x)) are well sorted providing overlapping is allowed.

5. The syntactic form of characteristic literals

In the previous sections we argued that it is sometimes useful to allow characteristic literals to appear in formulae. In this section we will briefly mention three standard representations for characteristic literals and propose a fourth one for use in the many sorted logic Llama.

5.1. Characteristic literals using unary predicates

The most common representation for characteristic literals is the one we have been using thus far in this paper – i.e. as monadic predicates. This would seem to be a natural representation in many ways since a sort represents a subsort of the universe of discourse and so does a monadic predicate.

5.2. Characteristic literals using the 'Isa' predicate

An alternative sometimes seen is to have a unique binary predicate symbol 'Isa'; thus e.g. we might write Isa(c,man) rather than Man(c). Characteristic resolution must be changed so that Isa(c,man) can be resolved against Isa(c,woman). Thus certain constant symbols representing sorts are treated specially by the unification algorithm.[9]

This seems perhaps a rather trivial syntactic modification but can be more convenient and expressive. One advantage is that one can now quantify over sorts and can therefore write formulae such as Isa(x,s) ∧ P(x,y) ∧ R(x,y) → Isa(y,s). It is also possible to generate an infinite number of sorts by relaxing the zero-adic nature of sort symbols. For example in the logic programming domain one might want to have the sorts number, string, list(number), list(string), list(list(number)), list(list(string)) ...[10] This can be achieved by making 'list' a rank one sort constructor. A term such as list(x) now represents infinitely many sorts. One possible problem with this idea is that some proofs of completeness of many sorted logics rely on there being a finite number of sorts.

A question which arises is what the sort of sorts is (because sort symbols are now terms and terms must have sorts). One can introduce a special sort (called, say, ss) which is the sort of any sort symbol or expression. In fact one could specialise this sort and create subsorts of it in order to distinguish different kinds of sorts (e.g. sorts with a natural ordering (such as number) from those with no natural ordering (such as colour)). Alternatively one could introduce mutually disjoint subsorts of ss, called s0 and sn, such that all rank zero sort symbols are of sort s0 and all others of sort sn. By then declaring a rank one sort constuctor 'linearlist' to take only arguments of sort s0 (rather than ss as would be usual if one wanted to have lists of anything) one could insist that lists were linear and expressions such as

Isa(c,linearlist(linearlist(number)))

would be ill-sorted. Similarly the term

lcons(lcons(4,nil),nil)

would be ill-sorted, where lcons is the linearlist constructor function whose sortal declaration is:

$\dot{lcons}$(x,linearlist(x)) = linearlist(x)

5.3. Characteristic literals using equality

Another method for representing sortal knowledge of terms within a sorted formulation other than by special purpose declarations is given in Walther [1987]. In order to express that a particular term α has a particular sort τ,

[9] The typed logic programming language LOGIN [Ait-Kaci and Nasr 1985] has such a unification algorithm (although it does not have an 'Isa' predicate). Every term is a type, there are no constant symbols; types at the bottom of the type hierarchy are effectively constant symbols; LOGIN also allows types to have attributes called *features* whose values may in turn be any type).

[10] Mycroft and O'Keefe [1984] have such a *parametrically polymorphic* type system.

one writes $\exists\beta{:}\tau\ \alpha{=}\beta$, i.e. one predicates that the term is equal to an existentially quantified variable of an appropriately restricted sort.

However, although one can express the information content of a characteristic literal using this technique (i.e. one can predicate that a term is of a particular sort), there is no obvious way to define a characteristic resolution rule for such literals in the way outlined for the previous two methods. So we hesitate to call this a technique for representing characteristic literals.

5.4. Characteristic literals using a binary 'Samesort' predicate

Llama is a *polymorphic* [11] logic, which means that predicate and function symbols can be overloaded to be defined on several different combinations of argument sorts; moreover in order to exploit this, variables in formulae do not have a unique sort but each formula has an associated *sort array* i.e. a set of constraints on the variables occurring in the clause. Furthermore, as already mentioned, Llama allows overlapping. The combination of these two features means that given the logic as formulated in Cohn [1987] it is possible for a clause to have several instances. The example given there is applying the substitution c/x to the unit clause Spouse(x,y) where the sort array[12] is {⟨Man,Woman⟩, ⟨Woman,Man⟩} and c is of sort Human; there are two instances:

$$\text{Man}(c) \rightarrow \text{Spouse}(c,y)$$
$$\text{Woman}(c) \rightarrow \text{Spouse}(c,y)$$

The first has the sort array {⟨Woman⟩} and the second the sort array {⟨Man⟩}. Although the search space is no worse than would be the case in unsorted logic,[13] it would be nice if the split in the search space could be avoided.

The proposal we make here achieves this; it makes use of another feature of Llama: the ability to describe the result sort of predicate symbols as being one of EE (i.e. ill-sorted), TT (i.e. definitely true), FF (i.e. definitely false) or UU (either true or false). We introduce a binary predicate 'Samesort' declared thus:

$\dot{\text{Samesort}}(\alpha,\alpha) = \text{TT}$
$\dot{\text{Samesort}}(\alpha,\beta) = \text{FF}$
where α and β are *base* sorts (i.e. immediately above $\perp$) and $\alpha{\neq}\beta$.

The instance of Spouse(x,y) referred to above can now be represented as the unique clause

$$\text{Samesort}(c,x) \rightarrow \text{Spouse}(c,y)$$

with the same sort array as its parent. The Samesort predicate is a logical predicate and a rule of characteristic resolution can be defined on it. If this representation is regarded as an internal representation for ordinary monadic characteristic literals then (assuming every sort has a complement) we only ever need negative (or alternatively positive) Samesort literals, each of which is of the form $\neg\text{Samesort}(\alpha,\beta)$ where β is a variable and α is not: a traditional characteristic literal of the form $\tau(\alpha)$ would be represented as $\neg\text{Samesort}(\alpha,\beta)$ where the sort array for the clause restricts the variable β to $\top\backslash\tau$

Two clauses containing the literals $\neg\text{Samesort}(\alpha_1,\beta_1)$ and $\neg\text{Samesort}(\alpha_2,\beta_2)$ respectively can be resolved together by unifying these two literals and forming a resolvent in the normal way but including the residue literal $\neg\text{Samesort}(\alpha,\beta)$ where α and β are the mgus of α_1 and α_2, and β_1 and β_2 respectively. The sort array for the resulting clause is created in a special way in that when 'diagonalizing' on β_1 and β_2 (i.e. substituting β_1 for β_2 in the sort array), the least upper bound of the sorts of β_1 and β_2 are taken, rather than the greatest lower bound as would normally be the case.

For example, suppose we have the unit clauses S6(c) and S5(c). These would be represented as $\neg\text{Samesort}(c,x)$ and $\neg\text{Samesort}(c,y)$, with sort arrays {⟨S1⟩} and {⟨S2⟩} respectively, where we are using the sort structure in Figure 2. The resolvent is $\neg\text{Samesort}(c,x)$, with a sort array of {⟨S1⊔S2⟩} = {⟨S4⟩} which is the representation of S3(c) as should be expected. If the two parent clauses had been S3(c) and S2(c) the sort arrays for the 'Samesort clauses' would be {⟨S4⟩} and {⟨S5⟩} respectively, and the sort array for the resolvent would be {⟨⊤⟩}. Llama would determine this

---

[11] Polymorphism can be useful because it can reduce the search space by collapsing multiple branches (e.g. see Ohlbach and Schmidt-Schauss [1985] or Cohn [1985]. It is also useful because it makes the logic more expressive; e.g. in the logic being developed in Randell and Cohn [1989] the binary predicate 'In' can be defined so that it is only well sorted when the first argument is a point and the second a boundary, or the first argument a boundary and the second a non boundary region.

[12] Here, we represent a sort array as a set of n-tuples of sorts: each n-tuple gives an allowable assignment of sorts for the n variables in the clause (in alphabetic order). In each of these *sort environments* the clause is of sort UU. Environments for which the clause is TT do not need to be kept because the clause is tautologous and the tuple can be deleted for the same reason that ordinary tautologies can be deleted. Similarly, when any result is FF the clause is a contradiction (because variables are implicitly universally quantified). Therefore in order to represent the environments in which a clause is well sorted we only need keep those for which the clause is of sort UU.

[13] In unsorted logic we would have to represent the unit clause Spouse(x,y) as two clauses

$$\text{Woman}(x) \wedge \text{Man}(y) \rightarrow \text{Spouse}(x,y)$$
$$\text{Man}(x) \wedge \text{Woman}(y) \rightarrow \text{Spouse}(x,y)$$

Thus the split in the search space is already present.

literal to be contradictory (using its elementary evaluation rule) and there would end up being no residue literal. This is what is desired since S3(c) and S2(c) clash leaving no residue.

Another advantage of this representation is that it allows clauses to be *merged* when otherwise they might not be. Two clauses in Llama can be merged to a single clause whenever the text of the clauses is identical even if the sort arrays differ (a similar technique is possible with the many sorted logic of Chaminade [1988]). For example, the representation of the Salt and Mustard problem mentioned earlier can be reduced from 30 to 20 clauses when this representation is used. Of course, complexity of the search space is traded for size of the sort array – but computation with sort arrays should be more efficient than general purpose resolution; moreover the proof theoretic analysis of the logic is considerably simplified by this innovation. For example, clauses 22 and 41 when represented using the Samesort notation become

$$\neg\text{Samesort}(\text{Mill},x)\vee\neg\text{Samesort}(\text{Cole},y)$$

with sort arrays

$\{\langle\text{Neither},\top\backslash\text{Salt}\rangle\}$ and

$\{\langle\top\backslash\text{Mustard},\text{Oneof}\rangle\}$

respectively. Because the text of these clauses is identical, they can be merged into a single clause by combining their sort arrays: $\{\langle\text{Neither},\top\backslash\text{Salt}\rangle, \langle\top\backslash\text{Mustard},\text{Oneof}\rangle\}$.

The modification to the existing implementation of Llama to incorporate 'Samesort' characteristic literals has not yet been completed so we are not able to report on the deduction statistics for the Salt and Mustard problem here.

If we were to allow arbitrary use of the Samesort predicate, rather than the restricted syntactic form described above, (i.e only negative literals, first argument always instantiated, second argument always a variable), then greater expressiveness can be obtained. For example, $\neg\text{Samesort}(x,y)$ (i.e. both arguments uninstantiated) will restrict the sorts of x and y so that they are not identical which cannot be easily stated otherwise. Even if nothing is known directly about the sorts of two constants c1 and c2 (i.e. they are both of sort $\top$), the unit clause $\neg$Samesort(c1,c2) could be used to give the information that their sorts are disjoint (or alternatively the same, if the negation sign is dropped). No investigation of how deduction is to be performed with these literals has taken place yet though.

6. A non substitutional framework for hybrid reasoning

Many Sorted Logic can be viewed as a particular kind of hybrid representation and reasoning system. Apart from many sorted logics, there are various proposals for hybrid representation schemes in the literature (e.g. KRYPTON [Brachman et al 1983], KL-TWO [Villain 1985], LOGIN [Ait-Kaci and Nasr 1985], Theory Resolution [Stickel 1985], REP [Allen and Miller 1986], and EcoNet [Miller and Schubert 1988]). A common theme is that different kinds of knowledge are represented separately and different reasoning procedures may be used on different kinds of knowledge. In the case of a many sorted logic, it is taxonomic knowledge which is being factored out. Usually, a criterion for deciding whether a particular kind of knowledge should be factored out of the general representation and treated specially is whether the special purpose reasoner remains decidable (the general purpose reasoner is unlikely to be so) and what the effect is on the computational complexity of the specialized reasoner (Levesque and Brachman 1987).

There are several hybrid reasoning systems which do not fit into Frisch's substitutional framework – e.g. Stickel's *Theory Resolution* or any many sorted logic which allows characteristic literals (such as Llama). In Frisch's model the box containing the special theory is only called when a substitution is proposed. In the more general framework we are developing here, the specialised box is called to perform other functions as well and thus the communication channel is much richer. In the case of a many sorted logic the specialised box will contain the sort structure and the declarations of the sortal behaviour of the non logical symbols. In any many sorted logic with characteristic literals the following questions may be asked of the specialised representation by the general representation:

1) Is this formula well sorted?
2) Is this substitution well sorted?
3) Do these characteristic literals (partially) clash and what is the residue if any?
4) Does this characteristic literal imply this one? (For use when checking clause subsumption).

In Llama the following additional questions can be asked (because of the use of the use of the **B** lattice for specifying the result sorts of predicate symbols):

5) Can the truth value of this literal be determined given these sortal constraints on the variables? (This implements what is called *evaluation* in Llama. It is closely related to procedural attachement and what Stickel [1985] calls unary Theory Resolution)
6) What can be determined about the sorts of the non variables terms in this literal, given these sort constraints on the variables? (This corresponds to the inference rule of *sortcasting* in Llama which, for example, given the declarations:

$$\ddot{\text{M}}\text{other}(\text{Woman},\text{Human}) = \text{UU}$$
$$\ddot{\text{M}}\text{other}(\text{Man},\text{Human}) = \text{FF}$$
$$\ddot{\text{c}}1() = \text{Human}$$

$\ddot{c2}() = \text{Human}$

allows one to deduce Woman(c1) from Mother(c1,c2)).

The task of formalizing this framework to cope with a reasonably wide class of theories is currently under investigation. Obviously what is required is a specification of an inference mechanism, a protocol for communication, and restrictions on the syntactic form of knowledge in both the specialised and the general theories that will allow metatheoretic results to be applicable to any logic which can be fitted to the framework.

7. Final comments

Allow characteristic literals in a many sorted logic is unusual if not controversial. This paper has discussed the use and representation of sortal literals. By allowing characteristic literals in a many sorted axiomatisation the logic becomes more expressive: more knowledge can be treated specially. A consequence of this is that computation can become more efficient. The price to be paid is the additional complexity of the inference machinery.

Acknowledgements

I am very grateful to all the people I have ever discussed many sorted logic with. Especial thanks in this case are due to Alan Frisch. My thanks also go to members of the AI group at Warwick: Ian Gent, Felix Hovsepian, David Randell and Guy Saward. I am indebted to my wife for her patience and understanding.

References

Ait-Kaci, H and Nasr, R [1985]"LOGIN: A Logic Programming Language with Built-in Inheritance," AI-068-85, Microelectronics and Computer Technology Corporation.

Allen, J F and Miller, B W [1986]"The HORNE Reasoning System in COMMON LISP," TR126 (Revised), University of Rochester.

Bartley, W W [1977]*Lewis Carroll's Symbolic Logic*, Harvester Press.

Brachman, R J, Fikes, R E, and Levesque, H J [1983]"Krypton: a functional approach to knowledge representation," *IEEE Computer*, vol. 16, no. 10, pp. 67-73.

Chaminade, G [1988]"Some Computational Aspects of an Order Sorted Calculus: Order Sorted Unification Using Compact Representation of Clauses," in *Proc ECAI*, pp. 625-630, Pitman.

Clarke, B L [1981]"A Calculus of Individuals Based on Connection," *Notre Dame Journal of Formal Logic*, vol. 22, no. 3, pp. 204-218.

Clarke, B L [1985]"Individuals and Points," *Notre Dame Journal of Formal Logic*, vol. 26, no. 1, pp. 61-75.

Cohn, A G [1983]"Mechanising a Particularly Expressive Many Sorted Logic," PhD Thesis, University of Essex.

Cohn, A G [1985]"On the Solution of Schubert's Steamroller in Many Sorted Logic," *Proc IJCAI 9*, pp. 1169-1174, Morgan Kaufmann, Los Altos.

Cohn, A G [1986]"Many Sorted Logic = Unsorted Logic + Control?," in *Research and Development in Expert Systems III*, ed. M Bramer, pp. 184-194, Cambridge University Press.

Cohn, A G [1987]"A More Expressive Formulation of Many Sorted Logic," *J Automated Reasoning*, vol. 3, no. 2, pp. 113-200.

Cohn, A G [1989]"Taxonomic reasoning with many sorted logics," *Artificial Intelligence Review*.

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

Frisch, A M [1986]"Parsing with Restricted Quantification: An Initial Demonstration," in *Artificial Intelligence and its Applications*, ed. A G Cohn & J R Thomas, John Wiley, Chichester.

Frisch, A M [1989]"A General Framework for Sorted Deduction: Fundamental Results for Hybrid Reasoning," in *Proc 1st Int. Conference on Knowledge Representation*, ed. R Brachman & H Levesque, Morgan Kaufmann, Los Altos.

Hayes, P J [1985]"Ontology of liquids," in *Formal Theories of the Commonsense World*, ed. J Hobbs and R Moore, pp. 71-107, Ablex.

Kowalski, R and Hayes, P J [1971]"Lecture Notes on Automatic Theorem Proving," DCL Memo 40, University of Edinburgh.

Levesque, H J and Brachman, R J [1987]"Expressiveness and Tractability in Knowledge Representation and Reasoning," *Computational Intelligence*, vol. 3, pp. 78-93.

Lusk, E and Overbeek, R A [1984]"The Automated Reasoning System ITP," Technical Report ANL 84 27, Argonne National Laboratory.

Lusk, E and Overbeek, R [1985]"Non Horn Problems," *J. Automated Reasoning*, vol. 1, pp. 103-107.

McDermott, D [1982]"A Temporal Logic for Reasoning about Processes and Plans," *Cognitive Science*, vol. 6.

McSkimin, J R and Minker, J [1977]"The Use of a Semantic Network in a Deductive Question Answering System," in *Proc IJCAI 5*, Morgan Kaufmann, Los Altos.

Miller, S A and Schubert, L K [1988]"Using specialists to accelerate general reasoning," in *Proc AAAI*, pp. 161-165, Morgan Kaufmann, Los Altos.

Mycroft, A and O'Keefe, R A [1984]"A Polymorphic Type System for Prolog," *Artificial Intelligence*, vol.

23, pp. 295-308.

Ohlbach, H J and Schmidt-Schauss, M [1985]"The Lion and the unicorn," *J. Automated Reasoning*, vol. 1, no. 3, pp. 327-332.

Randell, D and Cohn, A G [1989]"Modelling Topological and Metrical Properties of Physical Processes," in *Principles of Representation and Reasoning*, ed. R J Brachman, H J Levesque & R Reiter, Morgan Kaufmann, Los Altos.

Reiter, R [1981]"On the Integrity of Typed First Order Data Bases," in *Advances in Data Base Theory, Volume 1*, ed. H Gallaire, J Minker, J M Nicolas, Plenum Press.

Schmidt-Schauss, M [1988]"Computational Aspects of an Order Sorted Logic with Term Declarations," SEKI report SR-88-10, Universitaet Kaiserslautern.

Stickel, M E [1985]"Automated Deduction by Theory Resolution," *J Automated Reasoning*, vol. 1, pp. 333-355, Kluwer.

Stickel, M E [1986]"Schubert's Steamroller Problem: Formulations and Solutions," *J Automated Reasoning*, vol. 2, pp. 85-101.

Villain, M [1985]"The restricted language architecture of a hybrid representation system," in *Proc IJCAI*, pp. 547-551, Morgan Kaufmann, Los Altos.

Walther, C [1985]"A Mechanical Solution of Schubert's Steamroller by Many-Sorted Resolution," *Artificial Intelligence*, vol. 26, pp. 217-224.

Walther, C [1987]*A Many Sorted Calculus Based on Resolution and Paramodulation*, Pitman.

Towards a Theory of Access-Limited Logic for Knowledge Representation*

J. M. Crawford† and Benjamin Kuipers
Department of Computer Sciences
The University of Texas At Austin
Austin, Texas 78712
jc@cs.utexas.edu
kuipers@cs.utexas.edu

Abstract

One of the fundamental problems in the theory of knowledge representation is the difficulty of achieving both logical coherence and computational tractability. We present steps toward a theory of access-limited logic, in which access to assertions in the knowledge-base is constrained by semantic network style locality relations. Where a classical deductive method or logic programming language would retrieve all assertions that satisfy a given pattern, an access-limited logic retrieves all assertions reachable by following an available access path. The complexity of inference is thus independent of the size of the knowledge-base and depends only on its local connectivity. Access-Limited Logic, though incomplete, still has a well defined semantics and a weakened form of completeness ('Socratic Completeness') and is complete in some important special cases.

1 Introduction

Access-Limited Logic (ALL) is a logic for knowledge representation which utilizes semantic network style access limitations to guarantee computational tractability, even in very large knowledge-bases. Previous work has used the access limitations inherent in semantic networks for special purpose reasoning; in ALL these limitations form an integral part of the logic itself. A semantics for ALL has been defined by mapping queries, assertions and knowledge-bases to predicate calculus, and in terms of this mapping, consistency and weakened completeness results have been proven.

Reasoning is hard. If a knowledge representation language is as expressive as first-order predicate calculus then the problem of deciding what an agent implicitly knows (i.e. what an agent could logically deduce from its knowledge) is unsolvable. Thus a knowledge representation system, which does not give up expressive power, must use a weak inference system with an incomplete set of deduction rules or accept artificial resource limits (e.g. bounds on the number of applications of *modus ponens*). However, these approaches tend to be difficult to describe semantically and tend to place unnatural limits on an agent's reasoning ability [Levesque, 1986].

Our primary interest is the development of a system for the representation of commonsense knowledge. People seem to be able to reason efficiently with a very large commonsense knowledge-base. One reason for this is that when solving a given problem they only make use of the limited subset of their knowledge which is relevant to the problem.

Our approach in ALL begins with the well known mapping between atomic propositions in predicate calculus and slots in frames; the atomic proposition that the object a stands in relation r to the object b can be written logically as $r(a, b)$ or expressed, in frames, by including object b in the r slot of object a. Thus in a frame-based system it is natural to define the frames *directly accessible* from the frame a as those which appear in slots of a[1]. Extending this idea, one may define an *access path*, in a network of frames, as a series of frames each directly accessible from its predecessor. It proves useful to generalize this definition and allow access paths to branch on all values found in a given slot. A sequence of propositions defines an access path if any variable appearing as the first argument to a proposition has appeared previously in the sequence. For example, "John's parent's sister" can be expressed in ALL as the path:

$$(parent(John, x), sister(x, y))$$

*This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, UT-Austin. Research of the Qualitative Reasoning Group is supported, in part, by NSF through grant IRI-8602665, and by NASA through grants NAG 2-507 and NAG 9-200.

†Supported in part by a fellowship from GTE.

[1]Slots in ALL contain only frames and rules (defined below).

This defines an access path from the frame for *John* to the frames for *John*'s parents (found by looking in the parent slot of the frame for *John*), to *John*'s parents' sisters.

From access paths we build the inference rules of ALL. A rule is always associated with a particular slot in the network. Backward chaining *if-needed* rules are written in the form: $\beta \leftarrow \alpha$ (the structure of α and β is discussed below) and applied when a value for the slot is needed. Forward chaining *if-added* rules are written in the form: $\alpha \rightarrow \beta$ and applied when a new value for the slot is inserted. In either case the antecedent of the rule must define an access path (beginning with the slot the rule is associated with). For example, using the access path above we can write the if-needed rule:

$$aunt(John, y) \leftarrow parent(John, x), sister(x, y)$$

But note that we *cannot* write the (logically equivalent) rule:

$$aunt(John, y) \leftarrow sister(x, y), parent(John, x),$$

since the antecedent does not define an access path.

Where a classical deductive method or logic programming language would retrieve all known assertions that satisfy a given pattern, an access-limited logic retrieves all assertions reachable by following an available access path. The use of access paths alone, however, is insufficient to guarantee computational tractability in very large knowledge-bases. The evaluation of a path can cause an explosive back-chaining of rules which can spread throughout the knowledge-base. To prevent this, ALL introduces a second form of access limitation. The knowledge-base in ALL is divided up into partitions and back-chaining is not allowed across partitions — facts in other partitions are simply retrieved. When used together, these two kinds of access limitations can limit the complexity of inference to a polynomial function of the size of the portion of the knowledge-base accessible from the local partition.

However, a price must be paid for the efficiency of access limitations. Inference in ALL is weaker than inference in predicate calculus, since only locally accessible facts and rules can be used in deductions. However, any concept in the knowledge-base is potentially reachable; A string of queries, while conveying no new information, can move the focus of attention around to invoke the rules of the system in any order. Thus, access-limited logic has a property we call *Socratic Completeness* [2] — for any query of a proposition which is a consequence (in predicate calculus) of the knowledge-base, there exists a preliminary query after which the query succeeds. Further, ALL is *Partitionally Complete* — if the rules needed to derive a proposition are in the same partition as the proposition and the proposition can be proven using only backward-chaining rules then a query of the proposition succeeds.

The logical properties of ALL are stated more carefully in the next section. Section 3 examines the complexity of inference in ALL, section 4 presents a simple example from our implementation of ALL, section 5 discusses related work, and section 6 overviews our current plans for future work.

2 The Logical Coherence of ALL.

'Logical coherence' is an informally defined collection of desirable formal properties. We have proven that ALL has the following properties of a logically coherent knowledge representation system:

- ALL has a well defined syntax and proof theory.
- The semantics of ALL can be defined by a purely syntactic mapping of ALL knowledge-bases, queries and assertions to predicate calculus.
- In terms of this mapping, inference in ALL is consistent, Socratically Complete, and Partitionally Complete.

These properties are stated more precisely in theorems below.

We view these formal properties as necessary but not sufficient conditions for logical coherence. There remains, at least, the less formal claim that knowledge can be organized cleanly into partitions. This claim is discussed in the last subsection of this section.

The rest of this section sketches the formal development of ALL. The full account can be found in [Crawford and Kuipers, 1989].

2.1 Basic Notation

In the meta-theory of ALL we use the following notation. Quantified expressions are written in the form:

$$(\langle quantifier \rangle \langle variable \rangle : \langle range \rangle : \langle expression \rangle).$$

Thus, for example:

$$(\forall x : pred_1(x) : pred_2(x))$$

is read "For all x such that $pred_1(x)$, $pred_2(x)$". Similarly:

$$(\cup x : pred(x) : foo(x))$$

(where foo is a set valued function) denotes the union over all x such that $pred(x)$ of $foo(x)$.

If α is a list then:

- $head(\alpha)$ is the first element in α.
- $rest(\alpha)$ is all but the first element in α.

[2] The idea of Socratic Completeness was invented independently in [Powers, 1987] where it is referred to as Socratic Adequacy.

2.2 Syntax of ALL

We now build up the syntax of ALL. First the alphabet of an ALL is defined and then terms, propositions, access paths, rules, knowledge-bases, and finally ALL formula are defined.

2.2.1 Alphabets, Terms and Propositions

The *alphabet* of an Access-Limited Logic consists of countably infinite sets of variables, constants, and relations, the binary relation *isa*, the connectives $\leftarrow$ and $\rightarrow$, and the operators *query*, and *assert*. A *term* is a constant or a variable. A *proposition* is $r(t_1, \ldots, t_n)$ where r is an n-ary relation and all t_i are terms. A *fact* is a proposition such that all t_i are constants. For a proposition or list of propositions α:

- $vars(\alpha)$ is the set of variables appearing in α.
- $relations(\alpha)$ is the set of relations appearing in α.
- $constants(\alpha)$ is the set of constants appearing in α.

2.2.2 Access Paths

An *access path* (or simply a *path*) is a pair $\langle V, \alpha \rangle$ such that: V is a set of variables, and α is a list of propositions in which the first term of each proposition is either a constant, a member of V, or has appeared previously in α (this can be made precise by a simple recursive definition). If $V = \{\}$ then we omit it and say α is an access path. A path of length one is a *primitive* path.

2.2.3 Rules

$Conseq \leftarrow Ant$ is an *if-needed rule* iff:

- $Key = r(t_1, \ldots, t_n)$[3] is a proposition,
- $Conseq = Key$,
- Ant is a list of propositions,
- Either t_1 is a constant and Ant is a path, or t_1 is a variable and $\langle \{t_1\}, Ant \rangle$ is a path, and
- $vars(Conseq) \subset vars(Ant)$.

$Ant \rightarrow Conseq$ is an *if-added rule* iff:

- Key and $Conseq$ are propositions.
- Ant is a list of propositions such that $head(Ant) = Key$,
- $\langle vars(Key), Ant \rangle$ is a path, and
- $vars(Conseq) \subset vars(Ant)$.

For any rule ρ: $Key(\rho)$, $Conseq(\rho)$, and $Ant(\rho)$ access its respective components.

[3]Intuitively, the Key of a rule is the proposition that the rule is indexed under in the knowledge-base.

2.2.4 Knowledge-Bases

A *Knowledge-Base*, K, is a seven-tuple

$$\langle C, R, Nr, Ar, F, P, A \rangle.$$

The definition of a knowledge-base is given in figure 1. If

$$K = \langle C, R, Nr, Ar, F, P, A \rangle$$

is a knowledge-base and α is a proposition, list of propositions or a rule then α is *allowed* in K iff

$$constants(\alpha) \subset C \wedge relations(\alpha) \subset R.$$

2.2.5 Operations and Formula

If α is a path then $query(\alpha)$ is a *query*. If α is a primitive path then $query(\alpha)$ is a *primitive* query. If f is a fact then $assert(f)$ is an *assertion*. Any query or assertion is an *operation*. Any primitive query or assertion is a *primitive* operation. If $\mathcal{O} = query(\alpha)$ or $\mathcal{O} = assert(\alpha)$ is an operations and α is allowed in a knowledge-base K then $\mathcal{O}$ is *allowed* in K. If an operation $\mathcal{O}$ is allowed in a knowledge-base K then $\mathcal{O}(K)$ is an ALL *formula*.

2.3 Knowledge Theory

In this subsection we sketch the knowledge theory of ALL. The knowledge theory of ALL defines the value of ALL formula by defining the action of ALL operations (i.e. queries and assertions). Intuitively, the assertion of a fact f, adds f to a knowledge-base and returns the resultant knowledge-base (i.e. the knowledge-base after f is added and all applicable if-added rules are applied). A query of q, returns the substitutions needed to make q true in the knowledge-base, and a new knowledge-base (since processing the query may change the knowledge-base by invoking rules).

2.3.1 The Domain and Range of ALL Operations

Any given sets C, R, Nr, Ar, P and function A, define a finite set of possible knowledge-bases (differing only in facts) KB and an infinite set of ground substitutions Θ (binding variables in the alphabet to constants in C). For this subsection fix the sets C, R, Nr, Ar, P, and the function A. Then, for any operation, $\mathcal{O}$, allowed in the knowledge-bases in KB (note that an operation allowed in any knowledge-base in KB is allowed in all knowledge-bases in KB):

$$\mathcal{O} : KB \longrightarrow 2^{\Theta} \times KB.$$

We notate these returned values with pairs: $\langle$ < set of substitutions >, < knowledge-base > $\rangle$. and use *kb* and *sub* as accessors on their first and second components respectively.

A *Knowledge-Base*, K, is a seven-tuple $\langle C, R, Nr, Ar, F, P, A\rangle$ where:

C	=	A set of constants.
R	=	A set of relations.
Nr	=	A set of if-needed rules such that: $(\forall \rho : \rho \in Nr$: $constants(\rho) \subset C \wedge relations(\rho) \subset R)$.
Ar	=	A set of if-added rules such that: $(\forall \rho : \rho \in Ar$: $constants(\rho) \subset C \wedge relations(\rho) \subset R)$.
F	=	A set of facts such that: $(\forall f : f \in F$: $constants(f) \subset C \wedge relations(f) \subset R)$.
P	=	A set of *partitions*, subsets of $C \times R$, $\{p_1, \ldots, p_n\}$, such that: $(\cup i : 1 \leq i \leq n : p_i) = C \times R$ (i.e. each element of $C \times R$ is in some p_i).
A	=	A rule association function mapping: $Nr \cup Ar \Longrightarrow C \cup R$, such that: $(\forall \rho : \rho \in Ar \cup Nr : A(\rho) \in R \rightarrow \{A(\rho)\} = relations(Key(\rho)))$ (i.e. if a rule ρ is associated with a relation then that relation must be the one appearing in $Key(\rho)$).

Figure 1: Definition of a Knowledge-Base.

2.3.2 The Partitions of ALL Operations

Intuitively, a partition of K corresponds to a part of the knowledge-base which is somehow semantically cohesive and distinct from the rest of the knowledge-base. Facts and rules are often thought of as being 'in' partitions and operations are thought of as 'taking place' in subsets of $C \times R$ (unions of partitions). The intuition behind this comes from the frame view of ALL knowledge-bases. Recall that ALL constants can be thought of as frames and relations as slots in these frames (e.g. the fact $r(c_1, c_2)$ is equivalent to having the value c_2 in the r slot of the frame c_1). Thus a pair $\langle r, c\rangle$ can be thought of as a particular slot in a particular frame in the knowledge-base. We refer to such a pair as a *frame-slot*. Partitions are thus sets of frame-slots. Further, note that any primitive path α (by the definition of a path) must reference exactly one frame-slot and thus can be said to be 'in' a partition. In fact, since partitions can overlap, it can be in several partitions and any operation on α is performed 'in' the subset of $C \times R$ formed by taking the union of the partitions α is in. Intuitively, this union defines the rules which are available to the operation. Thus an operation on α has access to the rules of all partitions α is in.

More formally, if $K = \langle C, R, Nr, Ar, F, P, A\rangle$ is a knowledge-base and $\alpha = r(c, t_1, \ldots, t_n)$ is a primitive path (i.e. c a constant and all t_i, $1 \leq i \leq n$, are terms) and p is a partition of K then $\alpha \in p$ iff $\langle c, r\rangle \in p$. If $P = \{p_1, \ldots, p_n\}$ and $\mathcal{O} = query(\alpha)$ or $\mathcal{O} = assert(\alpha)$ then union of partitions for $\mathcal{O}$ is:

$$par_K(\mathcal{O}) = (\cup i : 1 \leq i \leq n \wedge \alpha \in p_i : p_i)$$

2.3.3 The Values of ALL Operations

Defining the values of ALL operations is primarily a mater of formalizing the action of forward and backward chaining rules. We use the following basic notation for knowledge-bases and substitutions: If $K_1 = \langle C, R, Nr, Ar, F_1, P, A\rangle$ and $K_2 = \langle C, R, Nr, Ar, F_2, P, A\rangle$ are knowledge-bases, then:

$$K_1 \cup K_2 = \langle C, R, Nr, Ar, F_1 \cup F_2, P, A\rangle.$$

If further, f is a fact allowed in K_1 then:

$$K_1 + f = \langle C, R, Nr, Ar, F_1 \cup \{f\}, P, A\rangle,$$

and $f \in K_1$ iff $f \in F_1$. If θ and η are substitutions then $\theta \circ \eta$ notates θ followed by η. If further, Θ_1 is a set of substitutions then $\eta \circ \Theta_1 = \{\eta \circ \theta_1 \mid \theta_1 \in \Theta_1\}$.

For a primitive operations $\mathcal{O}$, we define $\mathcal{O}_n(K, p)$ as the result of the operation $\mathcal{O}$ on the knowledge-base K, in some subset of $C \times R$, p, with rule chaining cut off at depth n (the full formal definition of $\mathcal{O}_n$ is given in [Crawford and Kuipers, 1989]). We then define $\mathcal{O}$ in terms of $\mathcal{O}_n$ as shown in figure 2. Note that since $\mathcal{O}$ is defined as the union over all n of $\mathcal{O}_n$, recursive rules (e.g. rules of form $q \leftarrow q$) do not cause any problems in ALL (or its lisp implementation). Figure 3 shows an example of a query on a simple knowledge-base.

2.4 Mapping ALL to Predicate Calculus

We define the semantics of ALL by mapping ALL knowledge-bases, assertions, and queries to (first order) predicate calculus. An alternative approach would be to define a model theory for ALL, in terms of which ALL is complete. This could be done, but we believe that (since the model theory of predicate calculus is well understood), mapping to predicate calculus and appropriately weakening the notion of completeness gives a more perspicuous picture of the semantics of ALL. Further, we believe that consistency and Socratic Completeness relative to predicate calculus (or perhaps an appropriate non-monotonic logic) are

If $\mathcal{O}$ is a primitive operation allowed in a knowledge-base K then:

$$\mathcal{O}(K) = (\cup n : n > 0 : \mathcal{O}_n(K, par_K(\mathcal{O})))$$

The result of a non-primitive operations is defined in terms of the results of its constituent primitive operations. Again assume that $\mathcal{O}$ is an operation allowed in K:

If $sub(query(q)(K)) = \{\}$ (i.e. $query(q)$ 'failed'),

$$\mathcal{O}(K) = \langle \{\}, K \rangle$$

else

$$\begin{aligned} \mathcal{O}(K) \quad = (\cup\theta \quad & : \theta \in sub(query(q)(K)) \\ & : \langle \theta \circ sub(query(\alpha'\theta)(K)), kb(query(q)(K)) \cup kb(query(\alpha')(K)) \rangle \ ) \end{aligned}$$

Figure 2: The definition of $\mathcal{O}$.

Assume $K = \langle C, R, Nr, Ar, F, P, A \rangle$ is a knowledge-base such that:

$$\begin{aligned} C &= \{c\} \\ R &= \{r_1, r_2\} \\ Nr &= \{r_1(c,x) \leftarrow r_2(c,x)\} \\ Ar &= \{\} \\ F &= \{r_2(c,c)\} \\ P &= \{\{\langle c, r_1 \rangle, \langle c, r_2 \rangle\}\} \end{aligned}$$

Further, $A(r_1(c,x) \leftarrow r_2(c,x)) = r_1$. Consider $query(r_1(c,x))(K)$ (where x is a variable). This is a primitive operation so we first compute $query_0(r_1(c,x))(K, par_K(r_1(c,x)))$. Rule back-chaining is cut off at depth 0 so no rules apply and $query_0(r_1(c,x))(K, par_K(r_1(c,x))) = \langle \{\}, K \rangle$ (an empty list of substitutions is returned since there is no known value of x such that the query succeeds). However when we calculate $query_1(r_1(c,x))(K, par_K(r_1(c,x)))$, the if-needed rule applies and $query_1(r_1(c,x))(K, par_K(r_1(c,x))) = \langle \{\{x/c\}\}, K + r_1(c,c) \rangle$ (where $\{x/c\}$ binds x to c). As n is increased further there are no other rules to apply so $query(r_1(c,x))(K) = \langle \{\{x/c\}\}, K + r_1(c,c) \rangle$.

Figure 3: A query on a simple knowledge-base.

Assume $K = \langle C, R, Nr, Ar, F, P, A \rangle$ is a knowledge-base such that:

$$\begin{aligned} C &= \{c\} \\ R &= \{r_1, r_2, r_3\} \\ Nr &= \{r_1(c,x) \leftarrow r_2(c,x)\} \\ Ar &= \{r_1(c,x) \rightarrow r_3(c,x)\} \\ F &= \{r_2(c,c)\} \\ P &= \{\{\langle c, r_1\rangle, \langle c, r_2\rangle, \langle c, r_3\rangle\}\} \end{aligned}$$

Finally, $A(r_1(c,x) \leftarrow r_2(c,x)) = r_1$, $A(r_1(c,x) \rightarrow r_3(c,x)) = r_1$. Consider $query(r_3(c,c))(K)$. This query must fail since $r_3(c,c)$ is not a fact in K and there are no if-needed rules for $r_3(c,c)$. But, any model of $\mathcal{PC}(K)$ must be a model of $\mathcal{PC}(r_3(c,c))$ (by the two rules and the fact that $r_2(c,c)$ is in F). Hence, inference in ALL is not complete.

Figure 4: A form of incompleteness in ALL.

Assume $K = \langle C, R, Nr, Ar, F, P, A \rangle$ is a knowledge-base such that:

$$\begin{aligned} C &= \{c\} \\ R &= \{r_1, r_2, r_3\} \\ Nr &= \{r_1(c,x) \leftarrow r_2(c,x), r_2(c,x) \leftarrow r_3(c,x)\} \\ Ar &= \{\} \\ F &= \{r_3(c,c)\} \\ P &= \{\{\langle c, r_1\rangle\}, \\ &\quad \{\langle c, r_2\rangle, \langle c, r_3\rangle\}\} \end{aligned}$$

Finally, $A(r_1(c,x) \leftarrow r_2(c,x)) = r_1$, $A(r_2(c,x) \leftarrow r_3(c,x)) = r_2$. Consider $query(r_1(c,c))(K)$. This query must fail since $r_2(c,c)$ is not a fact in K and is not in $par_K(r_1(c,c))$ (so no rules for $r_2(c,c)$ can fire). But, any model of $\mathcal{PC}(K)$ must be a model of $\mathcal{PC}(r_1(c,c))$ (by the two rules and the fact that $r_3(c,c)$ is in F).

Figure 5: Another form of incompleteness in ALL.

necessary properties for any knowledge representation system.

Mapping ALL to predicate calculus is fairly straight forward. Propositions do not change at all. Paths become conjunctions. Rules become implications with all variables universally quantified (there are some complications in mapping rules associated with frames (as opposed to slots) — these are discussed in [Crawford and Kuipers, 1989]). Knowledge-bases become the conjunction of their rules and facts. We notate the Predicate Calculus equivalent of an ALL object, a, by $\mathcal{PC}(a)$.

2.5 Consistency

Consistency is often intuitively thought of as "You can't derive a contradiction." Thus consistency requires that the substitutions returned by a query must be semantic consequences of the old knowledge-base. The requirements on the new knowledge base are more subtle. Consistency intuitively requires that propositions do not suddenly become true, or, in model theoretic terms, that models are not suddenly lost. Thus any model of the new knowledge-base must also be a model of the old knowledge base (and in an assertion a model of the formula being asserted):

Theorem 1 (Consistency) *For any knowledge-base K, any path α allowed in K, and any fact f allowed in K:*

1. $(\forall\theta \in \Theta \quad : \theta \in sub(query(\alpha)(K))$
   $\quad\quad\quad\quad\quad\quad : \mathcal{PC}(K) \models \mathcal{PC}(\alpha\theta))$
2. $\mathcal{PC}(K) \models \mathcal{PC}(kb(query(\alpha)(K)))$
3. $(\mathcal{PC}(K) \wedge \mathcal{PC}(f)) \models \mathcal{PC}(kb(assert(f)(K)))$

Proof (sketch): The proof of consistency is primarily a matter of carefully working through the definition of $\mathcal{O}$. We induct on n to show that $\mathcal{O}_n$ is consistent. We then induct on the length of α to show that $\mathcal{O}$ is consistent.

2.6 Completeness

Completeness can be thought of as "Any true fact is derivable." Thus completeness requires that all substitutions which are semantic consequences of the old knowledge-base are returned by query. Completeness also requires that true facts do not suddenly become false. In model theoretic terms this means that we do not gain models. Thus any model of the old knowledge-base must also be a model of the new knowledge-base. Note that the requirements for completeness are simply the requirements for consistency with their implications reversed:

Conjecture 1 (Completeness of ALL)
For any knowledge-base K, any path α allowed in K, and any fact f allowed in K, let Θ_α be the set of all ground substitutions binding all and only variables in α. Then:

1. $(\forall\theta \in \Theta_\alpha \quad : \mathcal{PC}(K) \models \mathcal{PC}(\alpha\theta)$
   $\quad\quad\quad\quad\quad\quad : \theta \in sub(query(\alpha)(K)))$
2. $\mathcal{PC}(kb(query(\alpha)(K))) \models \mathcal{PC}(K)$
3. $\mathcal{PC}(kb(assert(f)(K))) \models (\mathcal{PC}(K) \wedge \mathcal{PC}(f))$

Unfortunately, part one of this conjecture is false. In some cases, rules necessary for a query to succeed cannot be accessed. Two such cases are shown in the examples in figures 4 and 5. Notice, however, that in the example in figure 4:

$$query(r_3(c,c))(kb(query(r_1(c,c))(K)))$$

would succeed since $r_3(c,c)$ is added to

$$kb(query(r_1(c,c))(K))$$

by the if-added rule $r_1(c,x) \rightarrow r_3(c,x)$. Similarly, in the example of in figure 5:

$$query(r_1(c,c))(kb(query(r_2(c,c))(K)))$$

succeeds. This suggests the idea behind *Socratic Completeness*. Very informally, the Socratic Completeness Theorem says that for any query α which 'should' succeed in a knowledge-base, there exists a preliminary query β, after which a query of α succeeds. We also show a second type of partial completeness result, *Partitional Completeness*. Partitional Completeness says, that if all the information needed to process a query can be located by the if-needed rules in the partitions of the query, then that query succeeds.

2.6.1 Socratic Completeness

Theorem 2 (Socratic Completeness)
For any knowledge-base K, any path α allowed in K, and any fact f allowed in K, let Θ_α be the set of all ground substitutions binding all and only variables in α. Then:

1. $(\forall\theta \in \Theta_\alpha$: $\mathcal{PC}(K) \models \mathcal{PC}(\alpha\theta)$
   $\quad$: ($\exists\beta$: β *a path allowed in* K
   $\quad\quad$: $\theta \in sub(query(\alpha)(kb(query(\beta)(K)))))$
2. $\mathcal{PC}(kb(query(\alpha)(K))) \models \mathcal{PC}(K)$
3. $\mathcal{PC}(kb(assert(f)(K))) \models (\mathcal{PC}(K) \wedge \mathcal{PC}(f))$

Proof (sketch): Parts 2 and 3 follow relatively easily from the definitions of $\mathcal{O}$, and $\mathcal{PC}$. Part 1 is shown by induction on the length of α. The tricky part is the base case. We map K to an equivalent logic program $\mathcal{LP}(K)$. We show that for any rule in K which would apply on the next iteration of $T_{\mathcal{LP}(K)}$ (where T is the immediate consequence operator in logic programming — see [Crawford and Kuipers, 1989, Apt, 1988, Lloyd, 1984]) there exists a path in ALL the query of which causes the rule to fire. The result then follows by a completeness result for the study of logic programming.

2.6.2 Partitional Completeness

In order to state the partitional completeness theorem we first have to define which rules in the knowledge-base are considered 'part' of which partitions. A rule is considered a part of a partition if it can apply to a frame-slot in that partition. If p is a partition of a knowledge-base $K = \langle C, R, Nr, Ar, F, P, A\rangle$, and S is a set of rules from K then $S\backslash_p$ is the restriction of S to p (the set of rules from S which can apply to frame-slots in p — the formal definition is given in [Crawford and Kuipers, 1989]). If p is a union of several partitions then $S\backslash_p$ is just the union of S restricted to the partitions. The restriction of K to only the if-needed rules in p is:

$$K\backslash_p = \langle C, R, Nr\backslash_p, \emptyset, F, \{p\}, A'\rangle.$$

where A' is A restricted to the domain $Nr\backslash_p$. Note that the restriction of K to some union of partitions p is never computed (in the definition of ALL formula or in our lisp implementation of ALL), but is only a formal object used to state the partitional completeness theorem.

Theorem 3 (Partitional Completeness) *For any knowledge-base K, any primitive path α allowed in K, let Θ_α be the set of all ground substitutions binding all and only variables in α. Then:*

$$\begin{aligned}(\forall\theta \in \Theta_\alpha \quad &: \quad \mathcal{PC}(K\backslash_{par_K(\alpha)}) \models \mathcal{PC}(\alpha\theta)\\ &: \quad \theta \in sub(query(\alpha)(K)))\end{aligned}$$

Proof (sketch): The proof of this theorem again relies on results from the study of logic programming. Let $ground(\alpha)$ be the set of all variable free instantiations of α. Further, for any logic program pg, and any set of facts I, let:

$$\begin{aligned}T_{pg} \uparrow 0(I) &= I\\ T_{pg} \uparrow (n+1)(I) &= T_{pg}(T_{pg} \uparrow n(I))\end{aligned}$$

The key lemma is:

$$\begin{aligned}(\forall f \in ground(q) \quad &: \quad f \in T_{\mathcal{CP}(K\backslash_p)} \uparrow n(\emptyset)\\ &: \quad f \in kb(query_n(q)(K, par_K(q))))\end{aligned}$$

Which is shown by induction on n from the definition of $\mathcal{O}_n$ and which again implies the result by a completeness result from logic programming (for the induction to go through, this lemma must actually be strengthened somewhat — see [Crawford and Kuipers, 1989] for details).

2.7 About Partitions

An important part of the claim that ALL is logically coherent is the claim that knowledge can be divided into semantically distinct segments. Fortunately, partitions are not a new idea. Among other places, similar ideas can be found in Hayes' clusters [Hayes, 1985]. The related idea that reasoning can be done by separating rules into partitions is also not new. It is the idea behind, for example, blackboard architectures [Hayes-Roth, 1985] (the difference between partitions in ALL and the similar limitations in blackboard architectures is the idea of access paths, which allow us to use the entire knowledge-base as our 'blackboard').

3 The Computational Tractability of ALL.

In the worst case the time complexity of an ALL operation is a polynomial function of the size of the portion of the knowledge-base accessible from the local partition. We focus on primitive operations since non-primitive operations are defined as sequences of primitive operations (figure 2).

Assume $\mathcal{O}$ is a primitive operation allowed in a knowledge-base K. By examination of the rules in the partition of $\mathcal{O}$ in K we can determine:

- $reach(\mathcal{O}, K)$ — the set of all frame-slots which $\mathcal{O}$ can ever reference.
- $change(\mathcal{O}, K)$ — the set of frame-slots which $\mathcal{O}$ can ever change.
- $frames(\mathcal{O}, K)$ — the set of frames which $\mathcal{O}$ could possibly put into frame-slots in $change(\mathcal{O}, K)$.
- $operations(\mathcal{O}, K)$ — the set of all queries of frame-slots in $reach(\mathcal{O}, K)$ and assertions of frames in $frames(\mathcal{O}, K)$ into frame-slots in $change(\mathcal{O}, K)$.

(Formal definitions of these sets are given in [Crawford and Kuipers, 1989]). In a well partitioned knowledge-base these sets should be much smaller than the total size of the knowledge-base.

For a set S, let $|\,S\,|$ be the cardinality of S.

Theorem 4 (Complexity) *Assume $\mathcal{O}$ is a primitive operation allowed in a knowledge-base $K = \langle C, R, Nr, Ar, F, P, A\rangle$. Let*

- $o = |\,operations(\mathcal{O}, K)\,|$
- $c = |\,change(\mathcal{O}, K)\,|$
- $f = |\,frames(reach(\mathcal{O}, K))\,|$
- $r =$ *the number of rules in* $par_K(\mathcal{O})$.
- $a =$ *the maximum arity of any relation in* R.
- $v =$ *the maximum number of variables in any rule in* $par_K(\mathcal{O})$.

The worst case time complexity of calculating $\mathcal{O}(K)$ is bounded by:

$$a^2o^2r(r+f)^2c^{v+2}$$

Proof (sketch): Consider the vector of all operations $\mathcal{O}' \in operations(\mathcal{O}, K)$. For any n these operations produce a vector of knowledge-bases $\mathcal{O}'_n(K)$. We

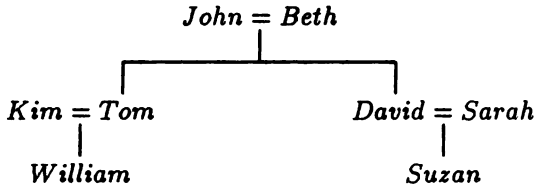


Figure 6: A genealogy.

show that if for some n and for all such $\mathcal{O}'$, $\mathcal{O}'_n(K) = \mathcal{O}'_{n+1}(K)$ then $\mathcal{O}(K) = \mathcal{O}_n(K)$. We then show that there must exist such an n which is less than or equal to $ao(r+f)c$ (by showing that knowledge-bases cannot shrink as n increases and showing a bound on how large they can grow). Finally, we show that the time to calculate any $\mathcal{O}'_n(K)$, from the values of all $\mathcal{O}'_{n-1}$, is bounded by $ar(r+f)c^{v+1}$.

4 Genealogy Example

An important part of our work with ALL has been our experience with the lisp implementation of ALL. We now present an introductory example from our implementation work. The knowledge-base consists of simple family relationships and the rules describe how to deduce more complex relationships. We emphasize that this example is one of the simplest we have implemented and is presented because it is relatively short and self-contained, yet gives a feel for the use of ALL and illustrates the use of access paths.

Figure 6 shows an example genealogy. To translate this into a knowledge-base assume that:

$$\begin{aligned} C &= \{ \ People, John, Beth, Kim, Tom, David, \\ &\quad Sarah, William, Suzan, Male, Female\} \\ R &= \{ \ isa, parent, child, son, daughter, brother, \\ &\quad sister, spouse, husband, wife, uncle, aunt, \\ &\quad cousin, gender\} \end{aligned}$$

Further, we need several if-added rules to enforce invariants in the knowledge-base. For example, we make sure that whenever there is a link $parent(x,y)$ there is also a link $child(y,x)$ (and vice-versa). Similarly, whenever there is a link $son(x,y)$ there are links $child(x,y)$ and $gender(x,Male)$, and so on. A important class of invariants are type restrictions — any frame put in a *parent*, *child*, or *spouse* slot '*isa*' '*People*'. The if-added rules are shown in figure 7. There are several other types of invariants which we

$$\begin{aligned} parent(x,y) &\rightarrow child(y,x) \\ child(x,y) &\rightarrow parent(y,x) \\ \\ son(x,y) &\rightarrow gender(y,Male) \\ son(x,y) &\rightarrow child(x,y) \\ daughter(x,y) &\rightarrow gender(y,Female) \\ daughter(x,y) &\rightarrow child(x,y) \\ \\ husband(x,y) &\rightarrow wife(x,y) \\ husband(x,y) &\rightarrow spouse(x,y) \\ husband(x,y) &\rightarrow gender(y,Male) \\ wife(x,y) &\rightarrow husband(x,y) \\ wife(x,y) &\rightarrow gender(y,Female) \\ spouse(x,y) &\rightarrow spouse(y,x) \\ \\ parent(x,y) &\rightarrow isa(y,People) \\ child(x,y) &\rightarrow isa(y,People) \\ spouse(x,y) &\rightarrow isa(y,People) \end{aligned}$$

Figure 7: If-added rules for genealogy example.

$$\begin{aligned} brother(x,z) &\leftarrow parent(x,y), son(y,z), x \neq z \\ sister(x,z) &\leftarrow parent(x,y), daughter(y,z), \\ &\quad x \neq z \\ uncle(x,z) &\leftarrow parent(x,y), brother(y,z) \\ uncle(x,z) &\leftarrow aunt(x,y), husband(y,z) \\ aunt(x,z) &\leftarrow parent(x,y), sister(y,z) \\ aunt(x,z) &\leftarrow uncle(x,y), wife(y,z) \\ cousin(v,z) &\leftarrow parent(v,w), parent(w,x), \\ &\quad child(x,y), y \neq w, \\ &\quad child(y,z), v \neq z \end{aligned}$$

Figure 8: If-needed rules for genealogy example.

could add (e.g. whenever there are links $child(x, y)$ and $gender(y, Male)$ then there is a link $son(x, y)$) but which are not necessary for this example. Similarly, we could add additional type restricting if-added rules for some of the more complex relations (e.g. $uncle(x, y) \rightarrow isa(y, People)$). Note, however, that the type restricting rules together with the other invariants ensure that any frames in the relations *son*, *daughter*, *husband*, and *wife* are '*People*'. We associate these if-added rules with the relations in their keys (e.g. $A(parent(x, y) \rightarrow child(y, x)) = parent$).

We use the if-needed rules shown in figure 8 to deduce the more complex relations. In these rules $x \neq y$ is true when x and y are bound to different frames. We associate these rules with the frame *People* (thus they are available to fill slots in any frame known to be a *People*). Notice that the rules for *uncle* and *aunt* are mutually recursive, but this causes no problem in ALL (though it would cause an infinite loop in Prolog) since *query* is defined as the union over all n of $query_n$ (see figure 2). Finally, we assume that the knowledge-base consists of a single partition, and initially contains no facts. We have thus defined an initial knowledge-base K_0.

Now we assert into K_0 the family relations in figure 6. This can be done by asserting the following path:

$$\begin{aligned} &wife(John, Beth), wife(Tom, Kim), \\ &wife(David, Sarah), son(John, Tom), \\ &son(Beth, Tom), son(John, David), \\ &son(Beth, David), son(Tom, William), \\ &son(Kim, William), daughter(David, Suzan), \\ &daughter(Sarah, Suzan) \end{aligned} \tag{1}$$

Asserting this path adds many more facts to the knowledge-base than just those mentioned in the path. For example, it adds $gender(Tom, Male)$, and that the frames *John*, *Beth*, *Tom*, *Kim*, *David*, *Sarah*, *William*, and *Suzan* are all *People*. Let K_1 be the knowledge-base after the assertion of the path in 1.

Finally, we can make queries into K_1. Consider first,

$$query(uncle(William, x))(K_1).$$

Assume,

$$p = par_{K_1}(uncle(William, x)).$$

Clearly,

$$query_0(uncle(William, x))(K_1, p)$$

fails. Similarly,

$$query_1(uncle(William, x))(K_1, p)$$

fails since the facts $parent(William, John)$ and $parent(William, Beth)$ are known, but no brothers of *John* or *Beth* are known. However,

$$query_1(brother(John, y))(K_1, p)$$

succeeds with y bound to *David* (by the if-needed rule for brother). Hence,

$$query_2(uncle(William, x))(K_1, p)$$

succeeds with x bound to *David*. Similarly,

$$query(cousin(Suzan, x))(K_1)$$

succeeds with x bound to *William*. One important advantage gained by the use of access paths is that the size of the knowledge-base could be increased with no effect on the time taken to compute these queries (unless we add frames which cause the access paths to branch — e.g. by adding more children of *John* and *Beth*).

5 Related Work.

ALL draws from several diverse fields and we will not have space here to examine in detail its relationship to the large body of previous work. We simply sketch in general terms the fields from which it draws and a few particularly relevant past approaches.

ALL draws from semantic networks [Findler, 1979, Brachman *et al.*, 1983, Bobrow and Winograd, 1985, Vilain, 1985] the intuition that retrieval and reasoning can be guided by the structure of the network. This has long been a key intuition behind semantic networks: "...the knowledge required to perform an intellectual task generally lies in the semantic vicinity of the concepts involved in the task." [Schubert, 1979]. ALL also draws from semantic networks its frame based data structures [Minsky, 1985].

ALL differs from past work on semantic networks in that it uses a single general purpose retrieval/reasoning mechanism which is guided by the structure of the network. Past work has generally used the structure of the network only for special purpose reasoning (spreading activation, classification etc.), and has relied on a first-order logic theorem prover [Brachman *et al.*, 1983, Schubert *et al.*, 1983] or a weaker deduction system [Levesque, 1984, Patel-Schneider, 1985, Vilain, 1985] for general reasoning.

A notable exception to this rule is the recent work of Schubert [Schubert, 1979, Haan and Schubert, 1986]. ALL and the networks of Schubert share several features including the use of access limitations to guide reasoning. The most obvious way to use the structure of a semantic network to limit access would be to perform deduction with facts not more than a few (say maybe two) nodes away in the network. The problem with this strategy is that some nodes (e.g. the node for your spouse) may have a large number of links, many of which are irrelevant to the problem at hand. The solution used in ECOSYSTEM is to maintain a taxonomy of knowledge and use this taxonomy

to guide reasoning [Haan and Schubert, 1986]. The difference in ALL is that access is limited to known *access paths*, which may access facts many nodes away in the network, but do so in a controlled fashion. Thus in ALL it is the structure of the knowledge itself (or more specifically the structure of the access paths in the rules) which controls access and reasoning.

The design of the inference mechanism in ALL has been heavily influenced by logic programming. In fact any function free logic program (without negation) can be written in ALL. Further, the notation, and the proofs of the completeness of logic programming [Apt, 1988, Lloyd, 1984], have been used extensively in the completeness proofs for ALL.

6 Discussion and Future Work.

Ultimately we are working towards a formal theory which has the expressive power of predicate calculus, and is consistent and Socratically Complete, but still has polynomial time complexity. The current formalism of ALL (unlike our Lisp implementation) can express only implication — not general negation. It is straight forward to add to ALL the ability to express full classic negation (i.e. not negation by failure), but then inference in ALL (using rules alone) is no longer Socratically Complete. For example, from two rules of the form:

$$\begin{aligned} p &\leftarrow q \\ \neg p &\leftarrow q \end{aligned}$$

one should be able to conclude $\neg q$, but neither rule can apply since there are no facts. We are currently working to increase the deductive power of ALL by adding a *Reductio Ad Absurdum* mechanism. This involves adding the ability to make an assumption and then reason about its consequences. If the consequences include 'false' then we can conclude the negation of the assumption. In the above example we assume q and derive p and $\neg p$. Thus we can conclude $\neg q$. We believe that such a mechanism will allow Socratically Complete reasoning in the presence of classic negation.

There is also no obvious way to express full existential quantification in our current formalism or our implementation. We have incorporated definite descriptions (e.g. "The man with the wooden leg"), which define a type of existential quantification (a definite description should pick out a unique known frame or, if there is no frame meeting the description, create a new one) into the implementation of ALL, but we have not yet formalized them as they do not seem to translate naturally into predicate calculus.

However, we have observed that commonsense reasoning often involves reasoning about groups of similar objects, and that much of this reasoning can be done without full first order quantification. One may, for example, reason about a large class of objects by reasoning about a representative object having the properties common to all objects in the group (or reason about a 'skolem' object having properties that some (unknown) object of the group is known to have). In our implementation work we have been developing a "commonsense set theory" entirely within the quantifier limitations of ALL, and have applied it to several examples.

In general, our lisp implementation of ALL is considerably ahead of our formalism. Beyond definite descriptions and common-sense set theory we have implemented full negation (using *Reductio Ad Absurdum* as discussed above) and an ability to make default assumptions. We have built up a small knowledge-base of common-sense knowledge and have investigated several classes of problems:

- We have written a version of the inferential distance rule of Touretzky [Touretzky, 1986]), and have looked at some standard examples of multiple inheritance (e.g. royal elephants and birds that are penguins) and at a Nixon diamond.
- We have implemented a fairly standard solution to the Yale shooting problems [Hanks and McDermott, 1986].
- Using our common-sense set theory we have implemented a solution to McCarthy's sterilization problem [1987] and other more complex problems involving sets of similar objects.
- To demonstrate Socratic Completeness and the use of *Reductio Ad Absurdum*, we have implemented a solution to the following logical puzzle taken from [Wylie, 1957]:

  > In a certain bank the positions of cashier, manager, and teller are held by Brown, Jones and Smith, though not necessarily respectively. The teller, who was an only child, earns the least. Smith, who married Brown's sister, earns more than the manager.
  > What position does each man fill ?

  Our solution involves the use of rules which define notions of partial orders and one-to-one relations between sets. When we state the problem in ALL, our lisp implementation initially fails to solve it, but after a suitable sequence of preliminary queries (essentially the questions one would ask a person to step them through the puzzle) is able to do so.

References

[Allen *et al.*, 1984] Allen, J. F., Giuliano, M., and Frisch A. M. *The HORNE Reasoning System*, TR 126, Computer Science Department, University of Rochester, Rochester NY., 1984.

[Apt, 1988] Apt, Krzysztof, R. Introduction to Logic Programming. To appear in *Handbook of Theoretical Computer Science*, ed. J. van Leeuwen., North Holland.

[Brachman and Levesque, 1985] Brachman, Ronald J. and Levesque, Hector, J. *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, Cal., 1985.

[Brachman *et al.*, 1983] Brachman, R.J., Fikes,R. E., and Levesque H. J. Krypton: a functional approach to knowledge representation, *Computer*, 16:67-73., 1983.

[Bobrow and Winograd, 1985] Bobrow, Daniel G., and Winograd, Terry. An Overview of KRL, a Knowledge Representation Language. In [Brachman and Levesque, 1985], pp. 263-285.

[Crawford and Kuipers, 1989] Crawford, J. M., and Kuipers, B. Access-Limited Logics. Forthcoming technical report.

[Findler, 1979] Findler, N.V., *Associative Networks: Representation and Use of Knowledge by Computer*, Academic Press, New York, 1979.

[Haan and Schubert, 1986] Haan, J., and Schubert, L. K. Inference in a topically organized semantic net. In *Proc. Natl. Conf. Am. Assoc. Artif. Intell., Philadelphia, Pa.*, 1986, pp. 334-338.

[Hanks and McDermott, 1986] Hanks, Steve, and McDermott, Drew. Default Reasoning, Nonmonotonic Logics, and the Frame Problem. In *Proc. Natl. Conf. Am. Assoc. Artif. Intell., Philadelphia, Pa.*, 1986, pp. 328-333.

[Hayes, 1985] Hayes, Patrick J. The Second Naive Physics Manifesto. In Hobbs, Jerry R. and Moore, Robert C., *Formal Theories of the Commonsense World*, Ablex Publishing Co.: New Jersey, 1985, pp. 18-30.

[Hayes-Roth, 1985] Hayes-Roth, B. A Blackboard Architecture for Control. In *Artificial Intelligence Journal*, 26:251-321, 1985.

[Kay, 1973] Kay, M. The MIND system. In *Natural Language Processing*, ed. R. Rustin., New York: Algorithmics Press, 1973.

[Levesque, 1986] Levesque, H. J. Knowledge representation and reasoning. In *Ann. Rev. Comput. Sci.* 1:255-87, 1986. Palo Alto, California: Annual Reviews Inc.

[Levesque, 1984] Levesque, H.J. A Logic of Implicit and Explicit Belief. In *Proc. Natl. Conf. Am. Assoc. Artif. Intell., Austin, Texas*, 1984, pp. 198-202.

[Lloyd, 1984] Lloyd, J.W. *Foundations of Logic Programming*, Springer-Verlag, New York, 1984.

[McCarthy, 1987] McCarthy, J. Generality in Artificial Intelligence. In *Communications of the ACM* 30:1030-1035, 1987.

[Minsky, 1985] Minsky, Marvin. A Framework for Representing Knowledge. In [Brachman and Levesque, 1985], pp. 246-262.

[Patel-Schneider, 1985] Patel-Schneider, P. A Decidable First-Order Logic for Knowledge Representation. In *Proc. Int. Jt. Conf. Artif. Intell., Los Angeles, California*, 1985, pp. 455-458.

[Powers, 1987] Powers, L. H. Knowledge by Deduction. In *The Philosophical Review*, LXXXVII, No. 3, 1978, pp. 337-371.

[Schubert, 1979] Schubert, L.K., R.G. Goebel, and N.J. Cercone, "The Structure and Organization of a Semantic Net for Comprehension and Inference," in [Findler, 1979], pp. 121-175.

[Schubert *et al.*, 1983] Schubert, L.K., Papalaskaris, M.A., and Taugher, J. Determining Type, Part, Color, and Time Relationships. In *IEEE Computer*, 16(10), 53-60, 1983.

[Touretzky, 1986] Touretzky, David S. *The Mathematics of Inheritance Systems*, Morgan Kaufmann, Los Altos, California, 1986.

[Vilain, 1985] Vilain, M. The restricted language architecture of a hybrid representation system. In *Proc. Int. Jt. Conf. Artif. Intell., Los Angeles, California*, 1985, pp. 547-551.

[Woods, 1975] Woods, W. What's in a link: foundations for semantic networks. In *Representation and Understanding: Studies in Cognitive Science* , ed. Bobrow, D. and Collins, A., New York: Academic, 1975, pp. 35-82.

[Wylie, 1957] Wylie, C.R., Jr. /em 101 Puzzles in Thought & Logic, Dover Publications Inc, Mineola, New York, 1957.

Solutions to a Paradox of Perception with Limited Acuity

Ernest Davis*
Courant Institute
New York, New York

Abstract

A realistic theory of perception and knowledge must deal with the fact that perceptions have limited acuity. However, in constructing such a theory, care must be taken to avoid a paradox whereby an agent can extract perfect knowledge from imperfect perceptions, using his knowledge of these imperfections. This paper presents two ways to avoid this paradox: first, by supposing that the agent has imperfect knowledge about his own perceptual system; second, by assuming that the perceptual system behaves nondeterministically.

1 Introduction

Consider the following inferences: (1) A driver cannot read a road sign from a mile off, even if the sign is in view. (2) A sundial cannot be used as a stop-watch, since the position of the shadow does not change perceptibly in a second. (3) The length of a table can be more accurately gauged by measuring it against a yardstick than by estimating by eye. These inferences rely on an understanding of the limits of perceptual acuity. A commonsense theory of perception must therefore be able to express these limits. The commonsense theory need not incorporate the detail suitable for evaluation of actual sensory systems ([Bajcsy et. al.,86], [Brooks, 87].) Rather, it should give a rough account, capable of supporting inferences such as those above.

An important characteristic of perceptual acuity is that a series of small changes, each imperceptible, may combine to form a perceptible change. For example, the shadow of a sundial moves imperceptibly in each second, but the total motion is visible after an hour. Formally, let us define two states of the world as "indistinguishable" (by the perceptions of an agent or by a measuring device) if the change from one to the other is imperceptible. Then the indistinguishability relation, though reflexive and symmetric, is not transitive. It is a *tolerance* on the space of world states [Hayes, 85] [Poston, 72].

This property of indistinguishability can lead to the following paradox: Let A, B, and C be three states of the world such that A is indistinguishable from B and B from C, but A is not indistinguishable from C. Suppose the real state of the world is A. Then the agent can see that the world is not in state C. If he knows enough about his own perceptual system, then, even though he cannot see that the state of the world is not B, he can infer it, since, if the world were in B then he could not see that the world is not in state C.

Consider, for example, a class of perceptions that varies along a single real-valued parameter, such as the measurement of temperature by a thermometer. Suppose that the perceptual system is accurate to within 2.0; that is, two states of the world are indistinguishable just if the values of the parameter differ by less than 2.0. If the actual value of the parameter is 10, then the agent sees that it is less than 12.0 and greater than 8.0, but he sees no greater detail. However, if he knows the laws governing his own perceptions, then he can deduce that the parameter's value of the parameter is exactly 10.0. But this is unrealistic.

There are at least two ways out of the paradox. The first is to deny that the agent knows the properties of his own perceptions accurately enough to carry out this deduction. The second is to deny that the perceptions of the agent depend deterministically on the state of the world. As an example of the latter, consider a thermometer whose readings are accurate only within a range of 2 degrees. In that case, a reading of 10 degrees implies only a real value between 8 and 12 degrees, no matter how well the thermometer is understood. (A third resolution, that perceptions are vague, is not easily expressed in a standard logic [Zadeh, 87] [Parikh, 83].)

Many natural theories of perception and knowledge implicitly contain assumptions leading to the above paradox. This paper will show how the theory presented in [Davis,88] gives rise to the paradox and will propose two possible modifications of the theory, corresponding to the above two resolutions of the paradox.

*Thanks to Leora Morgenstern for her suggestions. This research was supported by NSF grant #IRI-8801529.

2 Initial theory

Our initial theory [Davis, 88] combines a standard possible-worlds semantics for knowledge ([Moore, 80]) with a modified possible-worlds semantics for perception. To represent knowledge, we use a collection of possible worlds, also called *situations*. A situation is one possible state of the world at an instant. Situations are connected by knowledge accessibility relations. Situation $S2$ is accessible from situation $S1$ relative to agent A, written "k($A, S1, S2$)", if $S2$ is consistent with everything that A knows in $S1$. We can then represent the assertion "A knows Φ in $S1$" in the form "Φ is true in all situations $S2$ that is accessible from $S1$ relative to A." Situations are also organized temporally, but this will not be important here.

We represent perception using a reduced form of possible world, called a *layout*. A layout represents only the physical aspects of the world, since only these can be perceived. In our initial theory, layouts are connected by a relation of indistinguishability or *perceptual compatibility*. Layout $L1$ is perceptually compatible with layout $L2$ relative to agent A, written "pc($A, L1, L2$)", if $L2$ is consistent with everything that A can perceive in $L1$; that is, as far as A can see in $L1$, the world could be in $L2$. We represent facts about perception using constraints on compatible layouts. For example, the statement that John sees that object $O1$ and $O2$ abut in situation $S0$ can be expressed by asserting that, in each layout compatible with the layout of $S0$, $O1$ abuts $O2$

$$\forall_{L2}\ \text{pc(john,layout}(S0),L2) \Rightarrow \text{abuts(place}(O1, L2),\text{place}(O2, L2))$$

Here "layout(S)" denotes the layout contained in situation S, and "place(O, L)" is a function giving the position of object O in layout L.

We assume the following axioms:[1]

1. What is known is true. Formallly, knowledge accessibility is reflexive.

$$\forall_{A,S}\ \text{k}(A, S, S)$$

2. What is perceived is true. Formally, perceptual compatibility is reflexive.

$$\forall_{A,L}\ \text{pc}(A, L, L)$$

3. Any layout is part of some situation.

$$\forall_L \exists_S\ L{=}\text{layout}(S).$$

4. What is perceived is known. Formally, if a situation is consistent with A's knowledge, its layout is compatible with his perceptions.

$$\forall_{A,S1,S2}\ \text{k}(A, S1, S2) \Rightarrow \text{pc}(A,\text{layout}(S1),\text{layout}(S2))$$

5. If A perceives Φ then he knows that he perceives Φ.

$$\forall_{A,S1,S2,L3}\ [\text{k}(A, S1, S2) \wedge \text{pc}(A,\text{layout}(S2),L3)] \Rightarrow \text{pc}(A,\text{layout}(S1),L3)$$

This formal expression of (5), analogous to the standard axiom for positive introspection on knowledge, can be justified as follows. Consider the following equivalent statement:

$$\forall_{A,S1,L3}\ \neg\text{pc}(A,\text{layout}(S1),L3) \Rightarrow [\ \forall_{S2}\ \text{k}(A, S1, S2) \Rightarrow \neg\text{pc}(\text{layout}(S2),L3)\ ]$$

The embedded implication asserts that $L3$ is not compatible with the layout of any situation knowledge accessible from $S1$; that is, that A knows in $S1$ that he is not seeing $L3$. The antecedent of the overall statement is that A can see in $S1$ that the world is not in $L3$. Overall, therefore, the statement is that, if A sees something in $S1$ (some fact that rules out $L3$), then he knows that he sees it, which was the desired statement.

We can now display the paradox discussed in the introduction. Let $S1$ and $S2$ be situations, and let $L3$ be a layout compatible with the layout of $S2$ but not with the layout of $S1$. Then we have

$$\text{pc}(A,\text{layout}(S2),L3) \wedge \neg\text{pc}(A,\text{layout}(S1),L3)$$

The conclusion $\neg$k(A,$S1$,$S2$) follows directly from axiom (5); i.e. situation $S2$ is not consistent with A's knowledge in $S1$.

Consider now an intelligent agent equipped with a measuring device for a single ordered parameter. Let "param(L)" denote the value of the parameter in layout L. Assume the following conditions:

(i) For any layout L , there exist values $X1$ and $X2$ such that, for any layout $L2$, if param($L2$) $\leq X1$ or if param($L2$) $\geq X2$, then $L2$ is not compatible with L. That is, if the parameter value is far enough apart, then the layouts are distinguishable. Define the interval "p_range(L)" as the maximal interval not containing any such $X1$ or $X2$.

(ii) If $L1$ and $L2$ are compatible, and param($L1$) $<$ param($L2$) then p_range($L1$) contains values lower than any in p_range($L2$) and p_range($L2$) contains values greater than any in p_range($L1$).

It follows directly from these two assumptions that, for any $L1$, $L2$, if param($L1$) $\neq$ param($L2$), then there exists a layout $L3$ that is compatible with $L2$ but not with $L1$. Therefore, by axiom (5), no situation containing $L2$ is accessible from any situation containing $L1$. Thus, the agent knows the exact value of the parameter, since any situation with a different value is inaccessible.

Though (ii) is not a necessary feature of a compatibility relation, it is characteristic of many natural definitions of compatibility, and avoiding it may involve

[1] These axioms are not independent. (4) follows from {2,5}. (2) follows from {1,3,4}.

introducing artificial complications into the theory of perception.

3 First Solution

The first solution removes the assumption that an agent knows the rules governing his own perceptions. This assumption is built into the model above, in that compatibility is viewed as depending purely on the layouts and the agent. If agents are to be partially ignorant of the rules governing perception, these rules must vary from one possible world to another. We therefore redefine the compatibility relation "pc($A, L1, L2$)" to be a fluent, a term whose value varies from one world to another, rather than a predicate. We introduce the predicate "true_in(S, F)" to mean that fluent F is true in situation S. (Since "true_in" applies to fluents and not sentences, problems of self-reference do not arise.) The statement, "A sees in $S0$ that Φ is true," is expressed by stating that Φ is true in every layout $L2$ that is compatible with the layout of $S0$, using the compatibility relation of $S0$. For example, the sentence, "A sees in $S0$ that $O1$ abuts $O2$" is expressed as

$$\forall_{L2}\ \text{true\_in}(S0,\text{pc}(A,\text{layout}(S0),L2)) \Rightarrow \text{abut}(\text{place}(O1,L2),\text{place}(O2,L2))$$

Axioms 2, 4, and 5 now take the following forms:

2. $\forall_{A,S,L}\ \text{true\_in}(S,\text{pc}(A,L,L))$

4. $\forall_{S1,S2}\ \text{k}(A,S1,S2) \Rightarrow \text{true\_in}(S1,\text{pc}(A,\text{layout}(S1),\text{layout}(S2)))$

5. $\forall_{S1,S2,L3}\ [\text{k}(A,S1,S2) \wedge \text{true\_in}(S2,\text{pc}(A,\text{layout}(S2),L3))] \Rightarrow \text{true\_in}(S1,\text{pc}(A,\text{layout}(S1),L3))$

The paradox is now blocked, because the compatibility relation is relative to possible world. If in reality — that is, in some fixed possible world $S0$ — $L2$ is compatible with $L3$ but $L1$ is not compatible with $L3$, it does not follow from axiom (5) that a situation containing $L2$ is inconsistent with one containing $L1$. That conclusion can only be drawn if the compatibility of $L2$ with $L3$ is given relative to a world containing $L2$, and the incompatibility of $L1$ with $L3$ is given relative to world containing $L1$.

Indeed, we can construct models in which an agent can have substantial knowledge of the laws of perception, and still not know more about the value of a parameter than he can directly sense. For each real $D \in [0,1]$ and each real P, define a possible world $S_{D,P}$ satisfying the following:

i. $\text{param}(\text{layout}(S_{D,P})) = P$

ii. $\text{true\_in}(S_{D,P},\text{pc}(A,L1,L2)) \Leftrightarrow \text{param}(L2) - \text{param}(L1) \in [-D, 1-D]$

The knowledge accessibility relation is defined by requiring two worlds to be knowledge accessible just if the agent sees the same possible range for the value of the parameter in each.

iii. $\text{k}(A, S_{D1,P1}, S_{D2,P2}) \Leftrightarrow P1 - D1 = P2 - D2$

In this model axioms (1)-(5) are satisfied, as are the axioms of positive and negative introspection on knowledge; the agent knows the statement, "There exists a $D \in [0,1]$ such that layouts $L1$ and $L2$ are compatible only if $\text{param}(L2) - \text{param}(L2) \in [-D, 1-D]$," since this statement is true in all possible worlds; but, in world $S_{P,D}$, the agent knows only that the value of the parameter is in the range $[P-D, P+1-D]$, since each such value is attained in some accessible world.

4 Second Solution

Our second solution is modeled on a measuring device that exhibits non-deterministic error. Even with a perfect understanding of this device, an agent cannot extract more information out of it than its error allows.

We introduce the concept of a *percept,* which is the state of an agent's entire perceptual field at an instant. Any given physical layout may give rise to one of a range of possible percepts. In the thermometer example, the layout is the actual parameter value; the percept is the report of the device. The predicate "image(A, L, P)" means that layout L can give rise to percept P in agent A. Two layouts are compatible if they have some percept in common.

$$\text{pc}(A,L1,L2) \Leftrightarrow \exists_P\ \text{image}(A,L1,P) \wedge \text{image}(A,L2,P)$$

In a specific situation S, an agent A receives exactly one percept, denoted "percept(A, S)". If A perceives percept P in situation S, then he knows that the current layout is some layout with image P. Therefore, the statement "A sees that Φ holds in S" is expressed by saying that Φ holds in every layout that can give rise to the percept of S. For example, the statement "In $S0$, A sees that $O1$ abuts $O2$" is expressed as follows:

$$\forall_L\ \text{image}(A,L,\text{percept}(A,S0)) \Rightarrow \text{abut}(\text{place}(O1,L),\text{place}(O2,L))$$

Axioms (2), (4), and (5) are expressed as follows:

2. $\forall_{A,S}\ \text{image}(A,\text{layout}(S),\text{percept}(A,S))$

That is, the percept received in S is always possible for the layout of S.

4. $\forall_{A,S1,S2}\ \text{k}(A,S1,S2) \Rightarrow \text{image}(A,\text{layout}(S2),\text{percept}(S1))$

That is, for $S2$ to be consistent with $S1$, it must be possible for $S2$ to give rise to the percept A receives in $S1$.

5. $\forall_{A,S1,S2}\ \text{k}(A,S1,S2) \Rightarrow \text{percept}(A,S1) = \text{percept}(A,S2)$

That is, an agent knows what his perceptions are in any situation. Note that this formulation is simpler and more natural than in either of the two previous theories.

The paradox is now blocked. It is possible to have $L3$ compatible with the layout of $S2$ and incompatible with the layout of $S1$ and still have $S2$ consistent with $S1$, as long as the percept actually received in $S2$ is not one of the percepts compatible with $L3$.

We can again construct a model in which the agent has strong knowledge about the limits of perception, and yet cannot deduce more about the layout that he can see. Consider the measurement of a single parameter, so that we can identify a percept with a real number. Let image(L, X) hold for a layout L and real X if X − param$(L) \in$ [-1,1]; thus, the measurement is accurate to with 1. For each $D \in [-1, 1]$ and each real P, let $S_{D,P}$ be a situation in which param(layout$(S_{D,P})$) $= P$ and percept$(A, S_{D,P}) = P + D$. Let k$(A, S1, S2)$ just if percept$(A, S1)$ = percept$(A, S2)$. This model satisfies axioms (1) through (5), and also the axioms of positive and negative introspection on knowledge; the agent knows that the image is always within 1 unit of the true value of the parameter; but, given a percept P, the agent only knows that the value of the parameter lies between $P - 1$ and $P + 1$.

The two approaches can be combined by taking the "image" relation defined in the second solution, and making it dependent on the possible world, so that agents can have imperfect knowledge of the image producing operation. In this theory, the statement "A sees in S that $O1$ abuts $O2$" would be expressed

$$\forall_L \text{ true\_in}(S,\text{image}(A, L,\text{percept}(A, S))) \Rightarrow \text{abut}(\text{place}(O1, L),\text{place}(O2, L))$$

5 Conclusions

Our original motivation for developing the above new theories was to avoid the paradox which made it possible for an agent to make unrealistically powerful deductions from his perceptions. Serendipitously, we found that these theories not only avoided the paradox in a natural way, but also led to more expressive or more elegant languages of perception and knowledge. Having established these framework, we are now able to proceed with the development of a specific theory of perceptual acuity in worlds richer than the toy, one parameter world that we have used as an example above.

6 References

R. Bajcsy, E. Krotkov, and M. Mintz, (1986), "Models of Errors and Mistakes in Machine Perception," Tech. Report MS-CIS-86-26, GRASP Lab 64, U. Penn. 1986.

R. Brooks, (1987), "Visual Map Making for a Mobile Robot," in M. Fischler and O. Firschein (eds.), *Readings in Computer Vision,* Morgan Kaufmann.

E. Davis, (1988), "Inferring ignorance from the locality of visual perception," *Proc. AAAI-88,* pp. 786-791.

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

R. Moore (1980), "Reasoning about Knowledge and Action," SRI Technical Note #191.

R. Parikh (1983), "The problem of vague predicates," in Cohen and Wartofsky (eds.), *Language, Logic, and Method,* Reidel Publishers, pp. 241-261.

T. Poston (1972), *Fuzzy geometry,* Ph.D. thesis, University of Warwick.

L. Zadeh (1987), "Commonsense and Fuzzy Logic," in N. Cercone and G. McCalla (eds.) *The Knowledge Frontier,* Springer-Verlag, pp. 103-136.

Temporal Constraint Networks*

Rina Dechter †, **Itay Meiri and Judea Pearl**
Computer Science Department
Cognitive Systems Laboratory
University of California, Los Angeles, CA 90024

Abstract

The paper extends classical methods of solving constraint satisfaction problems to include continuous variables, thus providing a framework for processing temporal constraints. In this framework, called Temporal Constraint Satisfaction Problem (TCSP), variables represent time points, and temporal information is represented by a set of unary and binary constraints, expressed as disjunctions of temporal intervals. We present algorithms for performing the following reasoning tasks: Finding all feasible times for the occurrence of a given event, finding all feasible relationships between two events, and generating one or more scenarios that are consistent with the information provided.

Two approaches to processing temporal constraints are presented: Decomposition and relaxation. The first decomposes a TCSP into several simpler TCSP's (STCSP), each solvable in polynomial time. The decomposition is managed systematically by a backtrack algorithm and can employ traditional enhancement schemes (e.g. forward-checking, backjumping, learning etc.) to prune the number of STCSPs processed. This method is complete, but has exponential worst case complexity. The relaxation method is an extension of the common path-consistency algorithm. We discuss its special features when applied to TCSPs, prove its convergence, and discuss open problems regarding its complexity and effectiveness.

* This work was supported in part by Air Force Office of Scientific Research, Grant #AFOSR 88 0177.

† Current affiliation, Computer Science Department, Technion, Haifa, Israel.

1. Introduction

Problems involving temporal constraints arise in various areas of computer science such as scheduling, program verification, and parallel computations. Recent research in common-sense reasoning [Hanks 1986; Shoham 1988], natural language understanding [Kahn 1977; Allen 1984], and planning [McDermott 1982], has attracted more attention into such problems from the AI community. Several formalisms for expressing and reasoning about temporal knowledge have been proposed, most notably, Allen's interval algebra [Allen 1983], Vilain and Kautz's point algebra [Vilain 1986], linear inequalities [Malik 1983; Valdes-Perez 1986], and Dean and McDermott's time map [Dean 1987]. Each of these representation schemes is supported by a specialized constraint-directed reasoning algorithm. At the same time, an extensive research has been carried out over the past years on problems involving general constraints [Montanari 1974; Mackworth 1977; Gaschnig 1979; Haralick 1980; Freuder 1982; Dechter 1987], yet, much of this work hasn't been extended to problems involving temporal constraints.

This paper presents a unified approach to temporal reasoning based on constraint network formalism. Using this formalism, we were able to develop:

1. A formal basis for various algorithmic schemes, permitting the analysis of their complexity and range of applicability.

2. An economical representation called a **minimal network**, which encodes all temporal relations between a pair of event points, including absolute bounds on time distances.

3. An efficient scheme of generating specific temporal scenarios, consistent with the given constraints.

We envision a temporal reasoning system to consist of a temporal knowledge base, a routine to check its consistency, a query answering mechanism, and an inference mechanism capable of discovering new information. The primitive entities in the knowledge base are **propositions** to which we assign temporal intervals, e.g., "I was driving a car" or "the book was on the table"; each interval representing the time period during which the corresponding proposition holds. The temporal information might be relative (e.g., "P_1 occurred before P_2"), or metric (e.g., "P_1 had started at least 3 hours before P_2 was terminated"). To express less specific information, disjunctive sentences may also be needed (e.g., "P_1 occurred during or after P_2"). We also allow references to absolute time (such as 4:00 p.m.), and to the duration of propositions (e.g., "P lasted at least two hours"). Given temporal information of this kind, we want to answer queries such as: Is it possible that proposition P holds at time t? What are the possible times at which a proposition P holds? What is the temporal relationship between two propositions P_1 and P_2?

There have been various suggestions of how to represent temporal information. If propositions stand for events, and each proposition P_i is associated with an interval $I_i = [a_i, b_i]$, then information about the timing of events can be expressed by means of constraints on the intervals or their associated beginning and ending points. Allen [1983] defined 13 possible relationships between any pair of intervals, and every constraint is specified as a list of the possible relationships between a pair of intervals. Since this leads to computational difficulties when trying to find all the feasible relationships between a pair of intervals, Vilain and Kautz [1986] suggested to express the information by means of constraints on the beginning and ending point of each interval. This approach gives rise to a polynomial time algorithm, but can handle only a limited class of problems. Recently, Ladkin and Maddux [1989] have proposed an algebraic approach to problems similar to those posed by Allen and Vilain and Kautz.

One of the requirement of our system is the ability to deal with metric information. Since both Allen's interval algebra and Vilain and Kautz's point algebra do not offer a convenient mechanism for dealing with such information, we take a different approach. We consider time points as the variables we wish to constrain, where a time point may be a beginning or an ending point of a temporal interval, as well as a neutral point of time unrelated to any interval (e.g., 4:00 p.m.). Malik and Binford [1983] and Valdes-Perez [1986] suggested to constrain the **temporal distance** between time points, i.e. if X_i and X_j are two time points, a constraint on their temporal distance is of the form $X_j - X_i \leq c$. This leads to a set of linear inequalities on the time points under consideration. In order to express compound constraints on temporal distances, we must also allow disjunctive sentences. Consider the following example:

Example 1: John goes to work either by car (30-40 minutes), or by bus (at least 60 minutes). Fred goes to work either by car (20-30 minutes), or in a carpool (40-50 minutes). Today John left home between 7:10 and 7:20, and Fred arrived at work between 8:00 and 8:10. We also know that John arrived at work about 10-20 minutes after Fred left home.

We wish to answer queries such as: "Is the information in the story consistent?", "Is it possible that John took the bus, and Fred used the carpool?", "What are the possible times at which Fred left home?", etc. Let P_1 be the proposition "John was going to work", and P_2 the proposition "Fred was going to work". P_1 and P_2 are associated with the intervals $[X_1, X_2]$, and $[X_3, X_4]$ respectively. The beginning and ending points of these intervals are also associated with meaningful time points. For instance, X_1 represents the time John left home while X_4 represents the time Fred arrived at work. Several temporal constraints are given in the story. From the fact that it takes John either 30-40 minutes or more than 60 minutes to get to work, the temporal distance between X_1 and X_2 is constrained by

$$30 \leq X_2 - X_1 \leq 40 \text{ or } X_2 - X_1 \geq 60.$$

Similar constraints apply to $X_4 - X_3$ and $X_2 - X_3$. Choosing $X_0 =$ 7:00 a.m., the fact that John left home between 7:10 and 7:20 imposes the constraint

$$10 \leq X_1 - X_0 \leq 20.$$

The constraint on $X_4 - X_0$ assumes a similar from.

This paper introduces a framework based on constraint satisfaction formalism for representing and processing such problems. Within this framework two solution methods are established. Section 2 presents the temporal constraint satisfaction model. Section 3 deals with a restricted, simpler TCSP (called STCSP), solvable in polynomial time. Section 4 solves the general TCSP through decomposition into several STCSPs, while Section 5 describes a relaxation algorithm applicable to the general TCSP. Section 6 provides a summary and concluding remarks.

2. The TCSP Model

The definitions of the temporal constraint satisfaction model are based on similar definitions for the general

Constraint Satisfaction Problem (CSP) [Montanari 1974]. A **Temporal Constraint Satisfaction Problem (TCSP)** involves a set of **temporal variables**, $X_1, \ldots, X_n$, representing time points, whose domains are the real numbers, R. Unary and binary constraints on these variables are expressed in terms of **temporal distances**. A **binary temporal constraint** T_{ij}, between the variables X_i and X_j, is a disjunction of intervals indicating the permissible values for the distance $X_j - X_i$. Namely, the set of intervals

$$\{(a_1,b_1), \ldots, (a_n,b_n) \mid a_k \le b_k\}$$

represents the constraint

$$T_{ij}\colon\ (a_1 \le X_j - X_i \le b_1) \vee \cdots \vee (a_n \le X_j - X_i \le b_n).$$

Using set notation T_{ij} can be defined as:

$$T_{ij} = \{(x_i,x_j) \mid x_i \in R, x_j \in R, \text{ and}$$

$$\exists k, 1 \le k \le n, a_k \le x_j - x_i \le b_k\}.$$

We assume that the constraints are always given in a **canonical form**, namely, all intervals are pairwise disjoint ($\forall i, 1 \le i \le n-1, b_i < a_{i+1}$). The set of intervals specifying the constraint T_{ij} is called the **interval representation** of T_{ij}, denoted by $I(T_{ij})$,

$$I(T_{ij}) = I_1 \cup \cdots \cup I_n,$$

where $I_k = [a_k,b_k]$. Interchangeably, $I(T_{ij})$ will also be written $\{I_1, \ldots, I_n\}$.

A **unary temporal constraint**, T_i, on temporal variable X_i is the disjunction of inequalities (given in its canonical form):

$$T_i\colon\ (a_1 \le X_i \le b_1) \vee \cdots \vee (a_n \le X_i \le b_n),$$

and is defined by:

$$T_i = \{x_i \mid x_i \in R, \text{ and } \exists k, 1 \le k \le n, a_k \le x_i \le b_k\}.$$

A unary temporal constraint can also be represented by an interval representation. When we refer to the domain of variable X_i, we usually mean the set of values that satisfy the unary constraint T_i.

A **network of binary temporal constraints** (a **binary TCSP**) consists of a set of temporal variables $X_1, \ldots, X_n$, and a set of unary and binary temporal constraints. Such a network can be represented by a directed **temporal constraint graph**, $G = (V,E)$, where the nodes $V = \{1, \ldots, n\}$, represent the variables $X_1, \ldots, X_n$, and an edge $(i,j) \in E$ indicates that constraint T_{ij} is specified. The edge can be labeled by the interval set $I(T_{ij})$. Each input constraint, T_{ij}, implies an equivalent constraint T_{ji}, however, the graph of a given set of constraints contains only the ones explicitly specified. For presentation uniformity and algorithmic simplicity, a special time point, X_0, is introduced to represent the "beginning of the world". Its value is assumed to be 0. This enables us to refer to the unary constraint T_i as a binary constraint T_{0i} having the same interval representation. The constraint graph of Example 1 is given in Figure 1. A tuple $X = (x_1, \ldots, x_n)$ is called a **solution**, if the assignment $X_i = x_i$ satisfies all the constraints. A value v of X_i is **feasible**, if there is a solution such that $X_i = v$. The set of feasible values of a variable is its **minimal domain**. The network is **consistent** if at least one solution exists.

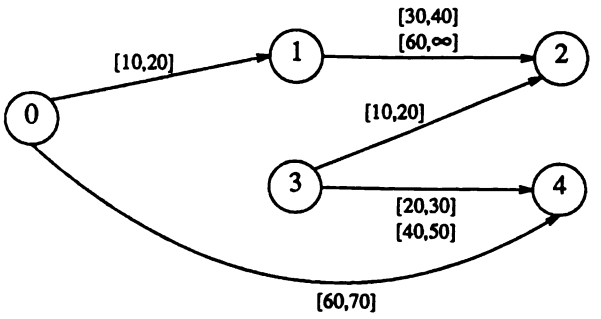


Figure 1. The constraint graph representing Example 1.

We define binary operations on temporal constraints: union, intersection, and composition, respecting their usual set-theoretic definitions. The **union** of two constraints T_{ij} and S_{ij}, denoted by $T_{ij} \cup S_{ij}$, admits only pairs of values that are allowed by either T_{ij} or S_{ij}. If $I(T_{ij}) = I_1 \cup \cdots \cup I_l$, and $I(S_{ij}) = J_1 \cup \cdots \cup J_m$, then

$$I(T_{ij} \cup S_{ij}) = I_1 \cup \cdots \cup I_l \cup J_1 \cup \cdots \cup J_m.$$

The **intersection** of two constraints T_{ij} and S_{ij}, denoted by $T_{ij} \oplus S_{ij}$, admits only pairs of values that are allowed by both T_{ij} and S_{ij}. If $I(T_{ij}) = I_1 \cup \cdots \cup I_l$, and $I(S_{ij}) = J_1 \cup \cdots \cup J_m$, then

$$I(T_{ij} \oplus S_{ij}) = K_1 \cup \cdots \cup K_n \ \ (n \le l + m),$$

where $\forall 1 \le k \le n$ there exist intervals $I_i, 1 \le i \le l$, and $J_j, 1 \le j \le m$, such that $K_k = I_i \cap J_j$. The **composition** of two constraints T_{ij} and T_{jk}, denoted by $T_{ij} \otimes T_{jk}$, admits only pairs of values (x_i,x_k), such that there is at least one value x_j, for which $(x_i,x_j) \in T_{ij}$ and $(x_j,x_k) \in T_{jk}$. If $I(T_{ij}) = I_1 \cup \cdots \cup I_l$, and $I(T_{jk}) = J_1 \cup \cdots \cup J_m$, then

$$I(T_{ij} \otimes T_{jk}) = K_1 \cup \cdots \cup K_n \quad (n \leq l \times m),$$

where $\forall 1 \leq k \leq n$, there exist intervals $I_i = [a,b]$, $1 \leq i \leq l$ and $J_j = [c,d]$, $1 \leq j \leq m$, such that $K_k = [a+c, b+d]$. Note that $\{K_1, \ldots, K_n\}$ may not be the canonical form.

Using these operations we define binary operations on networks of constraints. The union of two networks T and S, denoted by $T \cup S$, is defined as

$$\forall i,j, I((T \cup S)_{ij}) = I(T_{ij}) \cup I(S_{ij}).$$

Similarly, the intersection of two networks T and S, denoted by $T \cap S$, is defined as

$$\forall i,j, I((T \cap S)_{ij}) = I(T_{ij}) \cap I(S_{ij}).$$

A partial order among constraints can be defined in the following way: constraint T_{ij} is **tighter** than constraint S_{ij}, denoted $T_{ij} \subseteq S_{ij}$, iff every pair of values allowed by T_{ij} is also allowed by S_{ij}, i.e. for every interval $I_k \in I(T_{ij})$ there is an interval $J_l \in I(S_{ij})$ such that $I_k \subseteq J_l$. The tightest constraint between variables X_i and X_j is the **empty constraint**, denoted $\emptyset_{ij}$, where $I(\emptyset_{ij}) = \emptyset$. If the network contains an empty constraint, then it is trivialy inconsistent. The most relaxed constraint is the **universal constraint**, denoted U_{ij}, where $I(U_{ij}) = [-\infty, \infty]$. A partial order among networks of binary temporal constraints, having the same set of variables, can be defined as follows. A network T is tighter than network S, denoted $T \subseteq S$, iff the partial order is satisfied for all the corresponding constraints, i.e. $\forall i,j, T_{ij} \subseteq S_{ij}$. Two networks are **equivalent** iff they represent the same set of solutions. A network may have many equivalent representations. In particular there is one equivalent network which is minimal with respect to $\subseteq$, called the **minimal network** (note that the minimal network is unique because equivalent networks are closed under intersection).

Another important property of constraint networks is **decomposability**(1) [Montanari 1974]. A network is **decomposable** iff every assignment of values to any subset of k variables, $0 \leq k \leq n-1$, $S = \{X_{i_1}, \ldots, X_{i_k}\}$, that satisfies the constraints applicable to S (namely those involving only variables which are contained in S including the unary constraints), can be extended by an assignment of a value to any variable $X_{i_{k+1}} \notin S$, such that the extended assignment satisfies the constraints applicable to $S \cup \{X_{i_{k+1}}\}$. In particular, decomposability permits the construction of a solution by a **backtrack-free search**. A search is **backtrack-free** if every consistent assignment of a value to a variable is never changed because of a dead-ends in higher variables [Freuder 1982].

Given a network of binary constraints, the first interesting problem is to determine its consistency. If it is consistent, we may wish to find some solutions, each representing one possible scenario. Since it is hard to represent the infinite set of **all** possible solutions (i.e., the set of all scenarios) we settle for a more practical task of finding the minimal domains (answering queries such as "What are the possible times in which time point X_i could take place?"), and finding the minimal network (answering queries such as "What are all the possible relationships between a pair of time points?"). The rest of the paper discusses two approaches to solving these problems.

3. The Simple TCSP (STCSP)

A simple type of TCSP, called a **simple temporal constraint satisfaction problem (STCSP)**, is one whose all constraints have a single disjunct. This section presents a polynomial time algorithms for solving the STCSP. Section 4 extends the method to the general TCSP via a decomposition scheme.

Consider the constraint graph, $G = (V,E)$, of an STCSP T. Each edge $(i,j) \in E$ is labeled by a single interval $I(T_{ij}) = [a_{ij}, b_{ij}]$, representing the binary constraint:

$$a_{ij} \leq X_j - X_i \leq b_{ij}. \tag{1}$$

This constraint can be expressed as a pair of inequalities:

$$X_j - X_i \leq b_{ij}, \tag{2}$$

and

$$X_i - X_j \leq -a_{ij}. \tag{3}$$

Thus, solving the STCSP amounts to solving a set of linear inequalities on the variables $X_1, \ldots, X_n$ (called **simple linear program** in [Valdes-Perez 1986]).

This problem is well-known in the Operations Research literature, and it can be solved by the (exponential) simplex method [Dantzig 1962] or Khachiyan's algorithm [Khachiyan 1979], which is rather complicated and not efficient in practice. Fortunately, the special class of linear inequalities characterizing the STCSP admits a simpler, graph-based algorithm. Also, graph-based reasoning seems to better reflect human style of reasoning than linear programming does, and, hence, constitutes a more plausible model of qualitative reasoning. The graph representation we use, and which has been used by Shostak [1981], Aspvall and Shiloach [1980], Leiserson and Saxe [1983], and Liao and Wong [1983], also yields a

(1) In [Montanari 1974] decomposability is defined for minimal networks only.

simple criterion for deciding the consistency of STCSPs. A similar data structure called **time map** was introduced by Dean and McDermott [1987] but was not employed to solve the reasoning tasks addressed in this paper.

We associate an STCSP with a directed edge-weighted graph $G = (V,E)$, called **distance graph** (different than the constraint graph). Each node $i \in V$ represents the temporal variable X_i, and each edge $(i,j) \in E$ is labeled by a weight a_{ij}, and represents the linear inequality $X_j - X_i \leq a_{ij}$. In Example 1, if we assume that John used a car and Fred used a carpool, we get an STCSP having

$$I(T_{12}) = [30,40] \text{ and } I(T_{34}) = [40,50],$$

and the distance graph depicted in Figure 2.

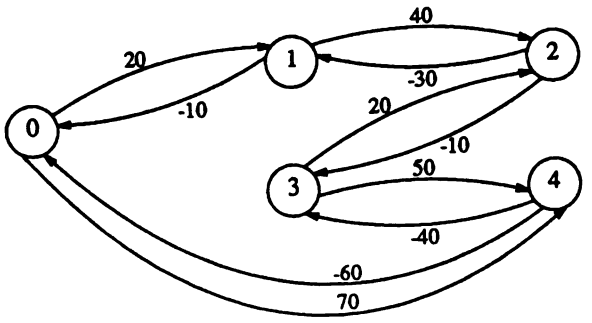


Figure 2. A distance graph representing one labeling of Example 1.

Each directed path, $i_0 = i, i_1, \ldots, i_k = j$, between nodes i and j in G, induces a constraint on the values of X_i and X_j, representing the composition of the constraints along the path:

$$X_j - X_i \leq \sum_{j=1}^{k} a_{i_{j-1},i_j}. \tag{4}$$

If there is more than one path between nodes i and j, then the intersection of the induced path constraints results in a new constraint on $X_j - X_i$. If we consider all the paths from node i to node j, then it can be easily verified that the induced constraint is:

$$X_j - X_i \leq d_{ij}, \tag{5}$$

where d_{ij} is the length of the shortest directed path from node i to node j. Based on this observation, Shostak [1981], Liao and Wong [1983], and Leiserson and Saxe [1983] presented the following theorem.

Theorem 1: An STCSP is consistent if and only if its distance graph has no negative directed cycles.

Proof. Suppose there is a negative cycle consisting of the nodes $i_1, \ldots, i_k = i_1$. If we sum the inequalities along the cycle we obtain $X_{i_1} - X_{i_1} < 0$ which cannot be satisfied. If there is no negative cycle in G, then the shortest directed path between each pair of nodes is well-defined. For all pair of nodes i and j, the shortest directed path from 0 to j satisfies:

$$d_{0j} \leq d_{0i} + a_{ij}, \tag{6}$$

or

$$d_{0j} - d_{0i} \leq a_{ij}. \tag{7}$$

Thus, the tuple $(d_{01}, \ldots, d_{0n})$, is a solution of the STCSP. □

The tuple $X_1 = (d_{01}, \ldots, d_{0n})$ assigns to each time point its latest possible time[2]. Similarly, the tuple that assigns to each time point its earliest possible time, can be found from

$$d_{i0} \leq a_{ij} + d_{j0}, \tag{8}$$

or

$$(-d_{j0}) - (-d_{i0}) \leq a_{ij}, \tag{9}$$

yielding $X_2 = (-d_{10}, \ldots, -d_{n0})$, as the desired solution.

From the preceding discussion it follows that an STCSP can be specified more effectively by a complete directed graph (i.e. each pair of variables has two directed arcs), where each edge (i,j) is labeled with the shortest path length d_{ij} of G. This graph is called d-graph and it corresponds to a more explicit specification of our STCSP (see Eq. (4) and Eq. (5)).

Theorem 2 (decomposability): Any consistent STCSP is decomposable relative to the constraints specified by its d-graph representation.

Proof. We have to show that any subset of instantiated variables S that satisfies all the shortest path constraints, is extensible by any other variable. If $S = \emptyset$, then any assignment of a value v to X_i, that satisfies the shortest path constraints T_{i0} and T_{0i} is a valid extension. v must satis-

(2) In fact, since it is a solution of Bellman's equations, it can be shown [Lawler 1976] that it is also a solution to the problem:

maximize $X_0 + X_1 + \cdots + X_n$

subject to: $X_0 = 0$ and $\forall i \neq j, X_j - X_i \leq a_{ij}$.

fy:

$$-d_{i0} \le v \le d_{0i}, \tag{10}$$

but, since all the cycles in G are nonnegative, $d_{0i} + d_{i0} \ge 0$, and there exists a value v satisfying Eq. (10). Assume now that S is not empty and let $S = \{X_1, \ldots, X_{k-1}\}$. Let $\{X_i = x_i \mid 0 \le i \le k-1\}$ be an assignment that satisfies the shortest path constraints among the variables in S. We need to find a value $X_k = v$ that satisfies the shortest path constraints T_{ik} and T_{ki} for $0 \le i \le k-1$. Thus, v must satisfy:

$$v - x_i \le d_{ik}, \tag{11}$$

$$x_i - v \le d_{ki}, \tag{12}$$

for every $0 \le i \le k-1$, or

$$v \le \min \{x_i + d_{ik} \mid 0 \le i \le k-1\} \tag{13}$$

$$v \ge \max \{x_i - d_{ki} \mid 0 \le i \le k-1\}. \tag{14}$$

Suppose the minimum is attained at i_0, and the maximum at j_0. Thus, v must satisfy:

$$x_{j_0} - d_{k,j_0} \le v \le x_{i_0} + d_{i_0,k}. \tag{15}$$

Since x_{i_0} and x_{j_0} satisfy the constraint between them, we have

$$x_{j_0} - x_{i_0} \le d_{i_0 j_0}. \tag{16}$$

This, together with $d_{i_0 j_0} \le d_{i_0,k} + d_{k,j_0}$, gives

$$x_{j_0} - d_{k,j_0} \le x_{i_0} + d_{i_0,k}. \tag{17}$$

Therefore, there exists v that satisfies the conditions in Eq. (15). □

The importance of Theorem 2 is that it provides an efficient algorithm for finding a solution for the STCSP; we simply assign to each variable any value that satisfies the d-graph constraints relative to previous assignments. Decomposability guarantees that such a value can always be found, regardless of the order of assignment. A second by-product of decomposability is that the domains characterized by the d-graph are minimal.

Corollary 1: Let G_d be the d-graph representation of a consistent STCSP. The set of feasible values for X_i is $[-d_{i0}, d_{0i}]$.

Proof. According to Theorem 2 any value $X_i = v$, $v \in [-d_{i0}, d_{0i}]$, can be extended to a solution. Thus, v is a feasible value. □

We have noted that the d-graph represents a tighter, yet equivalent, network of the original STCSP. From Theorem 2 we can now conclude that this new network is the minimal network.

Corollary 2: Given a consistent STCSP, T, the equivalent STCSP, M, defined by:

$$I(M_{ij}) = [-d_{ji}, d_{ij}], \forall i, j,$$

is the minimal network representation of T.

Proof. We will show that M is the minimal network by showing that it cannot be tightened any more, namely, that for any $d \in [-d_{ji}, d_{ij}]$, there exists a solution $X = (x_1, \ldots, x_n)$, such that $x_j - x_i = d$. Since the network is consistent then, according to Corollary 1, $X_i = d_{0i}$ is a feasible value. Clearly,

$$d_{0i} + d \le d_{0i} + d_{ij} \le d_{0j}, \tag{18}$$

and

$$d_{0i} + d \ge d_{0i} - d_{ji} \ge -d_{j0}. \tag{19}$$

Therefore, the assignment $\{X_i = d_{0i}, X_j = d_{0i} + d\}$ satisfies the constraints applicable to $\{X_i, X_j\}$, and by Theorem 2 it can be extended to a solution. □

Illustration: Consider the distance graph of Example 1 (Figure 2). Since there are no negative cycles, the corresponding STCSP is consistent. The shortest path distances, d_{ij}, are shown in Table 1. The minimal domains are $10 \le X_1 \le 20$, $40 \le X_2 \le 50$, $20 \le X_3 \le 30$, and $60 \le X_4 \le 70$. In particular, one special solution is the tuple $(d_{01}, \ldots, d_{04})$, namely $X_1 = 20$, $X_2 = 50$, $X_3 = 30$, $X_4 = 70$, which assigns to each time point the latest possible time. According to this solution John left home at 7:10 and arrived at work at 7:50, while Fred left home at 7:30 and arrived at work at 8:10. The interval representation of the minimal network is given in Table 2. Notice that the interval representation of the minimal network is symmetric in the sense that if $T_{ij} = [a, b]$ then $T_{ji} = [-b, -a]$. An alternative scenario, in which John used a bus and Fred used a carpool (i.e. $I(T_{12}) = [60, \infty]$ and $I(T_{34}) = [40, 50]$), results in a negative cycle and is therefore inconsistent.

The d-graph of an STCSP can be constructed by applying **Floyd-Warshall's** all-pairs-shortest-paths algorithm [Papadimitriou 1982] to the distance graph (see Figure 3). The algorithm runs to completion in time $O(n^3)$, and detects negative cycles merely by examining the sign of the diagonal elements d_{ii}. It constitutes, therefore, a polynomial time algorithm for determining the consistency of an STCSP, for finding one solution, and for determining both the minimal domains and the minimal net-

	0	1	2	3	4
0	0	20	50	30	70
1	-10	0	40	20	60
2	-40	-30	0	-10	30
3	-20	-10	20	0	50
4	-60	-50	-20	-40	0

Table 1. Lengths of shortest paths in the distance graph of Figure 2.

	0	1	2	3	4
0	[0]	[10,20]	[40,50]	[20,30]	[60,70]
1	[-20,-10]	[0]	[30,40]	[10,20]	[50,60]
2	[-50,-40]	[-40,-30]	[0]	[-20,-10]	[20,30]
3	[-30,-20]	[-20,-10]	[10,20]	[0]	[40,50]
4	[-70,-60]	[-60,-50]	[-30,-20]	[-50,-40]	[0]

Table 2. The minimal network corresponding to Figure 2.

work.

All-pairs-shortest-paths algorithm

1. for i:=0 **to** n **do** d_{ii}:=0;
2. for i,j:=0 **to** n **do** d_{ij}:=a_{ij};
3. for k:=0 **to** n **do**
  for i:=0 **to** n **do**
    for j:=0 **to** n **do**
      d_{ij}:=min$\{d_{ij}, d_{ik}+d_{kj}\}$;

Figure 3. Floyd-Warshall's algorithm.

4. A Decomposition Method for Solving the TCSP

One way of solving the general TCSP is to decompose it into several simple TCSPs, solve each one of them, then combine the results. Given a network of binary temporal constraints, T, we define a **labeling** of T as a selection of one interval from each temporal constraint (i.e. one disjunct). Each labeling defines an STCSP with an associated constraint graph whose arcs are labeled by the selected intervals. We can solve any of the TCSP's problems by considering all its STCSPs. Specifically, the original network is consistent iff there is a labeling whose associated STCSP is consistent. Any solution of T is also a solution of one of its STCSP and vice versa. Also the minimal network of T can be determined by the minimal networks associated with its individual STCSPs as stated in the following theorem.

Theorem 3: The minimal network M of a given TCSP satisfies $M = \cup_l M_l$, where M_l is the minimal network of the STCSP having the labeling l.

Proof. M is by definition the tightest of all networks equivalent to our TCSP, T. Therefore, to prove $M \subseteq \cup M_l$, it suffices to show that $\cup M_l$ is equivalent to T. However, this is clear form the fact that the solution set of T is identical to the union of the solution sets of its labelings. The opposite also holds. Since any solution in $\cup M_l$ is a solution to at least one M_l, its elimination would reduce the solution set of the corresponding STCSP. Moreover, since the solution sets offered by the labelings are disjoint, such an elimination reduces the solution set of T, and thus would violate the equivalence with the original TCSP. Therefore $\cup M_l \subseteq M$. □

Illustration: Consider Example 1. The interval representation of the minimal network is shown in Table 3. In this case only 3 out of the 4 possible labelings contribute to the minimal network.

	0	1	2	3	4
0	[0]	[10,20]	[40,60] [70]	[20,50]	[60,70]
1	[-20,-10]	[0]	[30,40] [60]	[10,30] [40]	[40,60]
2	[-70] [-60,-40]	[-60] [-40,-30]	[0]	[-20,-10]	[0,30]
3	[-50,-20]	[-40] [-30,-10]	[10,20]	[0]	[20,30] [40,50]
4	[-70,-60]	[-60,-40]	[-30,0]	[-50,-40] [-30,-20]	[0]

Table 3. The minimal network of Example 1.

The complexity of solving a general TCSP by generating all the labelings and solving them independently is $O(n^3k^e)$, where k is the maximum number of disjunct intervals of an edge, and e is the number of edges in the constraint graph (the number of input constraints). This enumeration process can be enhanced by viewing the problem as a higher-order constraint-satisfaction problem, where the variables are the arcs, their domains are their possible intervals, and then solving it using some back-

tracking techniques. Backtrack assigns an interval to an edge, as long as the condition of Theorem 1 is satisfied. If no such assignment is possible, it backtracks. Although the worst-time complexity of this approach is also $O(n^3k^e)$, it enables us to utilize enhancement techniques which, in practice, prove to substantially reduce the complexity of backtrack below its worst case value. Such techniques include backjump [Gaschnig 1979], variable ordering [Freuder 1982; Purdom 1983], value ordering [Haralick 1980; Dechter 1987], learning schemes [Dechter 1989] and various graph-based techniques [Dechter 1987; Dechter 1988].

5. A Relaxation Method for Solving the TCSP

The decomposition method for solving the TCSP suffers from two major drawbacks. First, even the more elaborate backtracking techniques we have suggested, do not seem to exploit the fact that most labelings differ from each other by only a small number of constraints. For each labeling we must apply the shortest-paths algorithm afresh. Second, the process of translating each labeling into a distance graph, although it takes only a polynomial time, might be too cumbersome in practice. Therefore, we have devised an alternative method applicable directly to the original constraint graph.

The all-pairs-shortest-paths algorithm can be considered a **relaxation** algorithm — in every step of the process the value of an edge is updated by an amount that depends only on the current values of adjacent edges. In fact, there is a rich family of similar algorithms [Aho 1974; Backhouse 1975; Lehmann 1977; Tarjan 1981a; Tarjan 1981b; Parker 1987], all based on the same principle. Montanari [1974] was the first to use such an algorithm, called **path consistency algorithm**, in the context of constraint satisfaction problems, and this was further explored and analyzed by Mackworth [1977], and Freuder and Mackworth [1985].

We present now a relaxation algorithm that attempts to enforce path consistency on TCSPs. The concepts of arc and path consistency for TCSPs are defined similarly to those of general CSPs [Montanari 1974; Mackworth 1977]:

Definitions: An arc (i,j) is **arc consistent** iff for any value $v \in T_i$, there is a value $w \in T_j$ such that $(v,w) \in T_{ij}$. A path through nodes $i_0, i_1, \ldots, i_m$ is **path consistent** iff for any pair of values $v_0 \in T_{i_0}$ and $v_m \in T_{i_m}$ such that $(v_0,v_m) \in T_{i_0 i_m}$, there exists a sequence of values $v_1, \ldots, v_{m-1}$, such that:

(a) $v_1 \in T_{i_1}, \ldots, v_{m-1} \in T_{i_{m-1}}$, and

(b) $(v_0,v_1) \in T_{i_0 i_1}, (v_1,v_2) \in T_{i_1 i_2}, \ldots, (v_{m-1},v_m) \in T_{i_{m-1} i_m}$.

A **network** is arc (path) consistent iff every arc (path) is arc (path) consistent.

Using $\oplus$ to denote intersection of constraints, and $\otimes$ to denote composition, Montanari's path consistency algorithm takes the form shown in Figure 4. The algorithm imposes a local consistency among triplets of variables i,j,k, until a fixed point is reached, or until some constraint becomes empty indicating an inconsistent network. For discrete-domain CSPs, Montanari showed that the algorithm terminates and that the resulting network is indeed path consistent.

Path consistency algorithm

1. for $i:=0$ **to** n **do** $A(i,i):=[0]$;
2. for $i,j:=0$ **to** n **do** $A(i,j):=I(T_{ij})$;
3. $B:=A$;
4. for $k:=0$ **to** n **do**
  for $i:=0$ **to** n **do**
    for $j:=0$ **to** n **do**
      $A(i,j):=A(i,j) \oplus A(i,k) \otimes A(k,j)$;
5. if $A \neq B$ **then goto 3**

Figure 4. A path consistency algorithm.

Comparing Figures 3 and 4, the path consistency algorithm is seen to be a generalization of the all-pairs-shortest-paths algorithm. When applied to an STCSP, the relaxation step that updates $A(i,j)$ amounts to two triangle operations of updating d_{ij} in Floyd-Warshall's algorithm. Therefore:

Theorem 4: Applying the path consistency algorithm to the constraint graph of an STCSP is identical to applying Floyd-Warshall's all-pairs-shortest-paths algorithm to its distance graph.

In general CSPs a path consistent network is not necessarily the minimal network. Montanari showed that when the constraints obey the distributivity property (i.e., that composition distributes over intersection), then any path consistent network is both minimal and decomposable. Moreover, in such a case only one application of the main loop (line 4) is sufficient for reaching the fixed point. When constraints are defined by one interval, the distributivity property holds. Indeed, for this case (the STCSP case), the path consistent network is minimal (Corollary 2), decomposable (Theorem 2), and requires only one iteration (see the shortest-paths algorithm). The question is whether any of these properties extends to

TCSPs whose constraints have multiple intervals. For instance, applying path consistency to Example 1 converges to the minimal network in a single iteration.

It turns out, however, that distributivity doesn't hold for the multi-interval TCSP. As an example consider the network of Figure 5. For convenience both direction of each edge are explicitly given. There are two paths from node 1 to node 3, representing the constraints $I(T_{13}) = [25,50]$, and $I(S_{13}) = [0,30] \cup [40,50]$, the later is obtained from composing T_{12} with T_{23}. Performing intersection first and then composition we get:

$$I(T_{01}) \otimes (I(T_{13}) \oplus I(S_{13})) =$$

$$([0,1] \cup [10,20]) \otimes ([25,30] \cup [40,50]) =$$

$$[25,31] \cup [35,70].$$

Performing composition first, and then intersection, results in:

$$(I(T_{01}) \otimes I(T_{13})) \oplus (I(T_{01}) \otimes I(S_{13})) =$$

$$([0,31] \cup [40,51] \cup [10,50] \cup [50,70]) \oplus$$

$$([25,51] \cup [35,70]) =$$

$$[25,70].$$

Clearly, distributivity doesn't hold. Indeed, if we apply the path consistency algorithm to this network, then one iteration gives $I(T_{2,3}) = [25,70]$, whereas in the minimal network (shown in Table 4), $I(M_{2,3}) = [25,31] \cup [35,70]$. Another application of the main loop results in a fixed point which is also the minimal network.

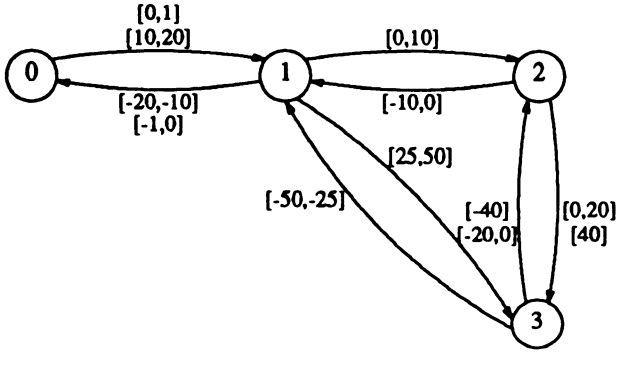


Figure 5. A nondistributive network.

	0	1	2	3
0	[0]	[0,1] [10,20]	[0,30]	[25,31] [35,70]
1	[-20,-10] [-1,0]	[0]	[0,10]	[25,30] [40,50]
2	[-30,0]	[-10,0]	[0]	[15,20] [40]
3	[-70,-35] [-31,-25]	[-50,-40] [-30,-25]	[-40] [-20,-15]	[0]

Table 4. The minimal network of Figure 5.

In view of the continuous domains of TCSPs one cannot guarantee a priori that path consistency terminates. It is clear, however, that running the algorithm indefinitely will results in a limit network. Each application of the main loop yields tighter, yet equivalent, network, and since the network is bounded below by M, a limit point is assured.

In order to analyze the applicability of the relaxation method the following questions need to be answered:

1. Is convergence guaranteed in a finite number of steps?
2. How many steps (if finite) are required?
3. Does the number of intervals on each edge proliferate?
4. Is the limit network minimal?

Although the answers to these questions are still unknown, we believe that in practice the path consistency algorithm converges efficiently to the minimal network for a large class of problems, and thus provides a practical alternative and a complementary approach to the decomposition scheme.

6. Summary and Conclusions

The paper provides a formal basis for dealing with problems involving temporal constraints. Using this formulation we can evaluate various algorithms for solving the fundamental problems. We presented two approaches to solving the TCSP. The decomposition scheme provides answers to all the basic tasks but its computational efficiency might be limited in practice. We have indicated

how decomposition can be improved by traditional constraint satisfaction techniques, such as backjumping, clustering, learning, and directional algorithms. The second approach amounts to using a path consistency (or relaxation) algorithm. The completeness and efficiency of this method are still open problems. We believe it is a good practical approach that will prove useful at least as an approximation.

In this paper we have not addressed some of the problems raised by Allen and Vilain and Kautz, primarily because their formulation does not involve metric information. Still, some relationship can be established. For example, we showed that the STCSP can be solved in time $O(n^3)$, the same time needed for Vilain and Kautz's algorithm for point algebra. On the other hand, Allen's interval algebra cannot be expressed using binary relations between time points. For example, the sentence "P_1 happened during P_2" imposes a constraint on four time points that cannot be decomposed into binary constraints (with or without multiple intervals). The natural extension of this work, therefore, is to explore higher-order TCSPs. In such a network, a constraint will involve more than two time points. For example the constraint "John drives to work at least 30 minutes more than Fred does", will be expressed by a 4-ary constraint: $X_2 - X_1 + 30 \leq X_4 - X_3$. Another extension is to invoke both logical and algebraic constraints.

References

[Aho 1974] Aho A.V., Hopcroft J.E. and Ullman J.D. In The Design and Analysis of Computer Algorithms. Reading, Mass: Addison-Wesley, Chapter 5, 1974.

[Allen 1983] Allen J.F. Maintaining Knowledge about Temporal Intervals. *CACM*, 26(no. 11): 832-843, 1983.

[Allen 1984] Allen J.F. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23:123-154, 1984.

[Aspvall 1980] Aspvall B. and Shiloach Y. A Polynomial Time Algorithm for Solving Systems of Linear Inequalities With Two Variables per Inequality. *SIAM Journal of Computing*, 9(no. 4):827-845, 1980.

[Backhouse 1975] Backhouse R.C. and Carre B.A. Regular Algebra Applied to Path-Finding Problems. *J. Inst. Math. Applications*, 15:161-186, 1975.

[Dantzig 1962] Dantzig G.B. *Linear Programming and Extensions*. Princeton: Princeton University Press, 1962.

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

[Dechter 1987] Dechter R. and Pearl J., Network-Based Heuristics for Constraint Satisfaction Problems. *Artificial Intelligence*, 34(no. 1):1-38, 1987.

[Dechter 1988] Dechter R. and Pearl J. Tree-Clustering Schemes for Constraint Processing. *Proc. of the Natl. Conf. on AI (AAAI-88), St.* Paul, 150-154, 1988.

[Dechter 1989] Dechter R. Enhancement Schemes for Constraint Processing: Backjumping, Learning, and Cutset Decomposition. To appear in Artificial Intelligence, 1989.

[Freuder 1982] Freuder E.C. A Sufficient Condition of Backtrack-Free Search. *JACM*, 29(no. 1):24-32, 1982.

[Gaschnig 1979] Gaschnig J. Performance Measurement and Analysis of Certain Search Algorithms. Technical Report CMU-CS-79-124, Carnegie-Mellon Univ., Pittsburgh, PA, 1979.

[Hanks 1986] Hanks S. and McDermott D.V. Default Reasoning, Nonmonotonic Logics, and The Frame Problem. *Proc. Natl. Conf. on AI (AAAI-86)*, 328-333, 1986.

[Haralick 1980] Haralick R.M. and Elliott G.L. Increasing Tree Search Efficiency for Constraint Satisfaction Problems. *Artificial Intelligence*, 14:263-313, 1980.

[Kahn 1977] Kahn K. and Gorry G.A. Mechanizing Temporal Knowledge. *Artificial Intelligence*, 9:87-108, 1977.

[Khachiyan 1979] Khachiyan L.G. A Polynomial Algorithm in Linear Programming. *Soviet Mathematics Doklady*, 20:191-194, 1979.

[Ladkin 1989] Ladkin P.B. and Maddux R.D. On Binary Constraint Networks. Kestrel Institute Technical Report, 1989.

[Lawler 1976] Lawler E.L. *Combinatorial Optimization: Networks and Matroids. New* York: Holt,

Rinehart and Winston, Chapter 3, 1976.

[Lehmann 1977] Lehmann D.J. Algebraic Structures for Transitive Closure. *Theoretical Computer Science*, 4:59-76, 1977.

[Leiserson 1983] Leiserson C.E. and Saxe J.B. A Mixed-Integer Linear Programming Problem Which is Efficiently Solvable. *Proc. of 21st Annual Allerton Conf. on Communications, Control, and Computing*, 204-213, 1983.

[Liao 1983] Liao Y.Z. and Wong C.K. An Algorithm to Compact a VLSI Symbolic Layout with Mixed Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and* Systems. CAD-2(2):62-69, 1983.

[Mackworth 1977] Mackworth A.K. Consistency in Networks of Relations. *Artificial Intelligence*, 8(no. 1):99-118, 1977.

[Mackworth 1985] Mackworth A.K. and Freuder E.C. The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems. *Artificial Intelligence*, 25(no. 1):65-74, 1985.

[Malik 1983] Malik J. and Binford T.O. Reasoning in Time and Space. *Proc. of IJCAI-83*, 343-345, 1983.

[McDermott 1982] McDermott D.V. A Temporal Logic for Reasoning about Processes and Plans. *Cognitive Science*, 6:101-155, 1982.

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

[Papadimitriou 1982] Papadimitriou C.H. and Steiglitz K.S. *Combinatorial Optimization*. Englewood Cliffs, NJ:Prentice-Hall, 1982.

[Parker 1987] Parker D.S. Partial Order Programming. Technical Report CSD-870067, UCLA, 1987.

[Shoham 1987] Shoham Y. *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence*. Cambridge: MIT Press, 1987.

[Shostak 1981] Shostak R.E. Deciding Linear Inequalities by Computing Loop Residues. *JACM*, 28(no. 4):769-779, 1981.

[Tarjan 1981a] Tarjan R.E. A Unified Approach to Path Problems. *JACM*, 28(no. 3):577-593, 1981.

[Tarjan 1981b] Tarjan R.E. Fast Algorithms for Solving Path Problems. *JACM*, 28(no. 3):594-614, 1981.

[Valdes-Perez 1986] Valdes-Perez R.E. Spatio-Temporal Reasoning and Linear Inequalities. Artificial Intelligence Laboratory, AIM-875, MIT, Cambridge, MA, 1986.

[Vilain 1986] Vilain M. and Kautz H. Constraint Propagation Algorithms for Temporal Reasoning. *Proc. Natl. Conf. on AI (AAAI-86)*, 377-382, 1986.

Impediments to Universal Preference-Based Default Theories

Jon Doyle*
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, Massachusetts 02139

Michael P. Wellman
USAF Wright Aeronautical Laboratories
AFWAL/TXI
Wright-Patterson AFB, Ohio 45433

Abstract

Research on nonmonotonic and default reasoning has identified several important patterns (autoepistemic, taxonomic specificity, chronological ignorance, threshold probabilistic) of nonmonotonic inference. The theories of reasoning based on each of these patterns may uniformly be viewed as theories of rational inference, in which the reasoner employs preferences among states of belief to select maximally preferred states of belief. Though research has identified some cases of apparent conflict between the preferences supported by different theories, it has been hoped that these special theories of reasoning may be combined into a universal logic of nonmonotonic reasoning. We show that the different categories of preferences conflict more than has been realized, and adapt formal results from social choice theory to prove that every universal theory of default reasoning will violate at least one reasonable principle of rational reasoning.

1 Introduction

The proliferation of formalisms for nonmonotonic inference [Ginsberg, 1987] attests to a diverse set of criteria for reasoning by default. These include circumscriptive inference [Lifschitz, 1986, McCarthy, 1980], which draws those conclusions valid in all minimal models of a set of axioms; autoepistemic inference [McDermott and Doyle, 1980, Reiter, 1980, Moore, 1985], which permits rules of inference to refer to unprovable statements as well as to provable ones; specificity-based taxonomic inference [Touretzky, 1986], which makes assumptions based on the most specific of the relevant prototypes; and chronologically ignorant inference [Shoham, 1988], which draws conclusions based on the shortest or simplest possible histories of events.

*Authors listed in alphabetical order. Jon Doyle is supported by National Institutes of Health Grant No. R01 LM04493 from the National Library of Medicine.

In addition to these major patterns, there are often domain-dependent reasons for adopting default policies in particular problem situations. In the absence of an explanation for why it exists, the proliferation of formalisms is unsatisfying. Our purpose in this paper is to investigate the natural question of whether there is some deeper or more comprehensive theory which combines or unifies all patterns (those known and those awaiting discovery) of nonmonotonic inference.

Toward this end, some theories have been proposed as unifications or partial unifications of some of these ways of making assumptions (see, for example, [Etherington, 1988, Konolige, 1987, Shoham, 1988]). At the same time, doubts about the existence of complete unifications have also been expressed, notably by Touretzky et al. [1987], who argue, for the special case of inheritance theories, that the gross differences between the theories stem from substantially different underlying intuitions about how to make assumptions. As they put it, the differing theories reflect a "clash of intuitions." These differences need not rule out the possibility of a unified theory encompassing inheritance, since a unified theory still may exist as long as these different intuitions can be shown to apply to disjoint cases of inferences. In that event, the unified theory can be obtained as a "big switch," or sum of the theories of each of the cases.

To investigate the existence of universal theories of default reasoning, we use Shoham's [1988] unifying formalism to translate questions about nonmonotonic inference into the context of rational decision making. If there is no irreconcilable clash in this more traditional context, the unified theory may be obtained by translating the decision-theoretic answer back to the case of nonmonotonic inference. Unfortunately, the translation illuminates an impediment to this unification, namely Arrow's [1963, 1967] celebrated results about the impossibility of universal social choice rules. We adapt Arrow's theory to obtain analogous results about the apparent nonexistence of universal theories of nonmonotonic and default reasoning, and draw on the literature of social choice to consider some possible ways around these results.

2 Preferential theories of default reasoning

The initial theories of default, circumscriptive, autoepistemic, chronologically ignorant, and specificity-based taxonomic inference had very different appearances. Despite their diversity, Shoham [1988] has shown how to cast each of these theories in similar form, as sound inference with respect to models minimal in some partial order. In his construction, a nonmonotonic logic is characterized by a strict partial order $\sqsubset$ over *interpretations*, which, depending on the base logical language $\mathcal{L}$, represent truth assignments, models, Kripke structures, or similar objects. The meaning of a nonmonotonic theory in these logics is then obtained by modifying the usual notions of satisfaction and entailment to take the model ordering into account. A model M *$\sqsubset$-satisfies* a formula P, written $M \models_{\sqsubset} P$, iff $M \models P$ and there is no $M' \sqsubset M$ such that $M' \models P$. A formula P *$\sqsubset$-entails* a formula Q, written $P \models_{\sqsubset} Q$, iff $M \models Q$ whenever $M \models_{\sqsubset} P$. Substitution of these variants for the usual satisfaction and entailment concepts yields a complete description of the nonmonotonic logic $\mathcal{L}_{\sqsubset}$.

Shoham illustrates the construction by providing partial orders corresponding to circumscription [Lifschitz, 1986], default logic [Reiter, 1980], the minimal knowledge logic of Halpern and Moses [1984], his own chronological ignorance [Shoham, 1988], and a few others, thus demonstrating that the construction accounts for a significant share of the extant formalisms. In circumscription, for example, models are ranked by minimality according to subset relations among extensions of specific predicates (i.e., abnormalities). That is, $M_1 \sqsubset M_2$ if the extension of the circumscribed predicate P in M_1 strictly contains its extension in M_2 and the two interpretations agree on all other functions and predicates. Other logics obtained in the same way include theories of inheritance (with models ordered according to the specificity of the defaults they satisfy), chronological ignorance (models ordered according to amount known about histories), and autoepistemic logic (models ordered according to a hierarchy among assumptions). All of these theories thus have the same formal structure, differing from each other only in how they order different models.

One natural interpretation of inference in this framework is as rational inference to maximally preferred states of belief, or to those conclusions that hold in all maximally preferred states of belief. Shoham's terminology is in accord with this interpretation, as he calls $\sqsubset$ a *preference* order, and the corresponding logical notions *preferential* satisfaction and entailment.[1]

[1] Given the view of inference as *max*imizing preferability of models, it seems unfortunate that Shoham retained the notation $\sqsubset$ and contrary sense of ordering from earlier treatments of circumscription, in which set-theoretic *min*imality of models was the key notion.

In fact, this way of viewing nonmonotonic inference is more than just an interpretation: it provides a justification for the formal structures of the various nonmonotonic logics missing from the early theories (but see [Doyle, 1983b, pp. 3-5]). The early theories provided clear formal concepts, but were less clear on why these concepts were interesting. The interpretation of default reasoning as rational inference, in which the agent adopts a belief if the expected utility of holding it exceeds the expected utility of not holding it, shows that the subjects of these theories are not completely new notions, but are instead cases of the concept made famous by Blaise Pascal and William James, whose ideas are commonly referred to as "Pascal's wager" and the "will to believe."

Pascal [1962] framed his problem of belief in God in the following way: he can either believe or doubt the existence of God, and God may either exist or not exist. If God exists and Pascal believes, he gains eternal salvation, but if he doubts he suffers eternal damnation. If God does not exist, belief may lead Pascal to forgo a few possible pleasures during his life that doubt would permit him to enjoy. We may summarize these evaluations in the decision matrix shown in Table 1, where ϵ represents the finite amount of pleasure enjoyed or forgone due to belief during Pascal's life. Of

	God exists	doesn't
Believe	$+\infty$	$-\epsilon$
Doubt	$-\infty$	$+\epsilon$

Table 1: Utilities of consequences in Pascal's decision about belief in God.

course, these same quantities modify the first column as well, but finite modifications to infinite quantities are negligible. As long as God's existence is not judged impossible, the expected utility of belief is $+\infty$, dominating the expected utility of doubt, $-\infty$. This convinced Pascal that doubt was not a viable alternative for him.

Much later, William James [1897] made the case that rational belief is ubiquitous in mundane reasoning. Today, his theory of the "will to believe" is one of the pillars of artificial intelligence practice, for knowledge representation and reasoning systems are filled with mechanisms for making assumptions in response to incomplete information. Taxonomic defaults, threshold probabilities, and nonmonotonic or circumscriptive proof procedures are all means used to guess at the information about actions and their consequences needed in deliberation. These mechanisms are ordinarily not presented in terms of rational choice, and their mechanization usually involves no decision-theoretic calculations. But when

closely examined, they are clearly based on rational responses to computational problems involving incomplete information (see [Doyle, 1983b, Doyle, 1989, Shoham, 1988]), as they avoid or minimize the costs associated with acquiring and analyzing the needed information, and are used until they prove wrong whenever the risks of serious consequences of error are judged to be low enough. Of course, Pascal's decision and other cases of adopting beliefs are usually viewed as making individual assumptions, where default rules make classes of assumptions. But we may assimilate the two cases theoretically by evaluating individual default rules as individual assumptions, that is, by adopting a default rule if the expected utility of using it to make individual assumptions is great enough.

3 Conflicting preferences about defaults

Since each of the specific sorts of nonmonotonic reasoning can be viewed as cases of rational inference, many have hoped or expected that with careful analysis one could combine the choices made in each into a single rational choice, yielding in effect a universal theory of nonmonotonic or default reasoning. However, the various theories are founded on different criteria for choice among belief states. The potential for conflict among them may impede any integration attempt.

For example, the famous "Yale shooting problem" of Hanks and McDermott [1987] illustrated that existing nonmonotonic logics are too weak to arbitrate conflicts among abnormality minimization of different individuals. In this view, the normality of loadedness after waiting and life after shooting are two conflicting criteria. Defenders of nonmonotonic logics have responded by proposing a third criterion—such as chronological minimization or some causality theory—to resolve the issue. However, as Hanks and McDermott point out, in some contexts other criteria (perhaps even chronological *max*imization for diagnostic reasoning) may be compelling, leading to further unresolvable conflicts. It seems a good bet that enterprising researchers will always be able to generate problems that fall through the cracks of fixed configurations of criteria.

In fact, numerous examples suggest that conflicts are unavoidable. The most widely known conflicts occur in inference from the most specific prototypes, where multiple dimensions of specificity within a taxonomic lattice can result in conflicting preferences between conclusions. An example is the famous "Nixon diamond" (so called because of the shape of its diagram when written as an inheritance network; perhaps also because it is so hard): Republicans are typically not pacifists, Quakers are typically pacifists, and Nixon is a Republican Quaker. The question is, is Nixon a pacifist or not? Since neither default is more specific than the other, one cannot tell simply from the information given in the taxonomic lattice. Moreover, though one might resolve the question of Nixon's pacifism empirically, one cannot expect to use that resolution to yield a rule which correctly interprets the large number of formally similar but substantially dissimilar conflicts among other taxonomic defaults.

Conflicts are also possible between pairs of more global preference orders. For example, chronological preferences can conflict with specificity preferences. Suppose we know that TRACK-STAR(t, x) implies FINISH-RACE$(t + n, x)$, while TRACK-STAR-ON-STEROIDS(t, x) implies FINISH-RACE$(t + n - d, x)$. The interpretation with a more specific causal antecedent leads to a result which is chronologically less ignorant. This example too is typical of a large number of structurally similar conflicts. For example, at some institutions computer science graduate students usually finish their doctorates in five or six years, but students of theoretical computer science usually finish in three or four, and students working on large computer systems can take seven or eight.

Similarly, ordering assumptions according to their relative probability can conflict with specificity orders. For example, suppose that with probability .99, over half of the kernels in a bag of microwave popcorn will pop within 2.5 minutes of starting the oven. But if we pick a pair of kernels from the bag, the probability that at least one of these kernels pops within 2.5 minutes is much less than .99. In this case, thanks to the law of large numbers, the more specific information leads to the less probable conclusion. This example too is simply one representative of a large class of similar conflicts.

Conflicts also occur because reasoners may have multiple informants or refer to multiple authorities to obtain their information. Most practical artificial intelligence systems are designed to incorporate all the available knowledge about the relevant subjects by combining expertise from multiple sources. This means that differences of opinion between experts must be worked out, either in advance or while reasoning. In the simplest case, one might consider encoding each expert's knowledge as a separate set of rules in the system, or as justifications for a subset of the rules which name the expert proffering them. In this case, as Thomason [1986] points out, conflicts between experts become conflicts within the expert system. Of course, the system designer can instead try to reconcile these conflicts at design time, but this may not always be feasible if some conflicts are too subtle to detect, or if the experts themselves knowingly hold mutually irreconcilable opinions. Thus if the system must perform in isolation from the original experts, one must expect it will sometimes have to deal with conflicts as they arise. For instance, many adults have had the experience of having to administer medications to them-

selves or to their children while on vacation, only to find that several medications have been prescribed by different doctors or for different symptoms, with each medication contraindicating the others.

4 Social choice and nonmonotonic logics

Each of the existing nonmonotonic formalisms was originally designed to capture a single criterion for preference among competing interpretations. In these formalisms the preferential nonmonotonic logics are direct expressions of the global preference criteria. To obtain a unified nonmonotonic logic, we must aggregate the several individual preference orders into a global order.

Constructing global preference orders can be difficult, since any comprehensive mechanism for nonmonotonic reasoning must embody some way of handling the conflicts that arise among the different patterns of inference. There are two major approaches taken to resolve conflicts: to choose to satisfy one preference instead of another, and to refuse to satisfy any of the conflicting preferences. In fact, each of the different theories proposed for nonmonotonic reasoning takes one of these approaches. For example, nonmonotonic logic, autoepistemic logic, default logic, and "credulous" inheritance [Touretzky *et al.*, 1987] describe how a single set of axioms and rules may yield several different, often incompatible sets of conclusions closed under inference. In these theories, conflicts between specific defaults are resolved in every possible way, with each maximal consistent set of preferences yielding a different set of conclusions. In contrast, circumscription, closed-world reasoning, Pollock's [1987] defeasible inference, and so-called "skeptical" inheritance [Horty *et al.*, 1987] resemble ordinary logic in that they describe how a set of axioms or rules yields a single set of conclusions closed under inference. These theories handle conflicts among preferences by failing to draw any conclusions in those circumstances.

Any universal theory of default reasoning must provide a rationale for its treatment of conflicts, whether credulous, skeptical, or sometimes one or the other. Placing responsibility for resolving potential conflicts on the human designer is infeasible because for large sets of criteria it is difficult to anticipate all of the potential conflicts. Furthermore, introduction of new criteria necessitates complete restructuring of the global preference order. A more satisfactory solution would exploit the concept of *modularity* to base conflict resolution mechanisms on general rules of combination that could be applied either manually or automatically as the need arises. As is widely recognized, modular design is critical to the successful construction of complex structures, and large commonsense knowledge bases certainly count as such.

To investigate this approach formally, we say that an *aggregation policy* is a function that specifies the global order corresponding to any given set of individual orders. Let $\sqsubset_i$ denote the preference order corresponding to the ith pattern of inference to be included in the unified logic. Each $\sqsubset_i$ could represent preferences based on a single criterion, such as predicate minimization, specificity, or chronological ignorance. Or, they might be at a finer grain, denoting the separate predicates to minimize, the individual dimensions of specificity, or individual default rules (as in Section 4.2). In any case, each $\sqsubset_i$ reflects a distinct attribute, encoding the local preferences over interpretations according to its dictates. Let I denote the set of all order indices. The multicriteria nonmonotonic logic problem is then to aggregate the orders $\sqsubset_i$: $i \in I$ into a global preference order $\sqsubset$.

For example, one simple aggregation function is unanimous decision: $M_1 \sqsubset M_2$ iff $M_1 \sqsubset_i M_2$ for all i that rank the two. This policy, of course, is extreme skepticism, failing to resolve any conflicts whatsoever.

Another sort of aggregation function comes from applying some kind of voting scheme, for example, majority rule among the criteria: $M_1 \sqsubset M_2$ iff

$$|\{i \in I \mid M_1 \sqsubset_i M_2\}| > |\{i \in I \mid M_2 \sqsubset_i M_1\}|.$$

Simple majority rule, however, is technically not a legal aggregation policy because the resulting global order $\sqsubset$ is not guaranteed to be transitive. (The intransitivity of majority rule is also known as "Condorcet's voting paradox," after the eighteenth century social scientist who discovered it [Roberts, 1976].)

This voting analogy can be taken quite literally. The problem of designing aggregation policies has been studied extensively in economics, under the heading *social choice theory*. In the language of social choice theory, the ranked interpretations $M_1, M_2, \ldots$ are *candidates*, the $\sqsubset_i$ are *individual orders*, and the global order is the *social ranking*. The aggregation policy itself is called a *social choice function*.

Reasoning has been viewed in social terms in AI by several authors. The most prominent example is Minsky's "society of mind" [Minsky, 1986], which explicitly models thinking as the aggregate activity of many small mental agents. Nowakowska [1973] explores the relation of social choice concepts to the psychology of agents made up of sets of motivational components. In the context of nonmonotonic reasoning, Borgida and Imielenski [1984] appeal to committee decision-making as a metaphor for default inference, and Doyle [1983a, 1985, 1988] presents nonmonotonic reasoning from a group decision-theoretic perspective.

The main result of social choice theory is a startling theorem due to Arrow [1963] that establishes the nonexistence of social choice functions possessing a set of desirable and apparently reasonable properties. In Sections 4.2 and 4.3, we show that slightly modified

versions of this result apply to preferential nonmonotonic logics, with important implications for the potential construction of universal default formalisms. We first discuss the hypotheses underlying these results.

4.1 Aggregation principles

The principled design of an aggregation policy for multicriteria preferences begins with a consideration of properties we think a reasonable policy should exhibit. The properties we propose are analogs of Arrow's desiderata for social choice, and can be expressed technically using Shoham's notation. We first present the proposed properties, and then discuss their desirability.

1. *Collective rationality.* The global order $\sqsubset$ is a function of the individual orders $\sqsubset_i$, which are unrestricted partial orders.
2. *Pareto principle (unanimity).* If $M_1 \sqsubset_i M_2$ for each $i \in I$, then $M_1 \sqsubset M_2$. The global order agrees with unanimous preferences.
3. *Independence of irrelevant alternatives (IIA).* The relation of M_1 and M_2 according to $\sqsubset$ depends only on how the $\sqsubset_i$ rank those two interpretations. That is, considering new alternatives does not alter rankings among the originals.
4. *Non-dictatorship.* There is no $i \in I$ such that for every M_1 and M_2, $M_1 \sqsubset M_2$ whenever $M_1 \sqsubset_i M_2$, regardless of the $\sqsubset_j$ for $j \neq i$. That is, there is no "dictator" whose preferences automatically determine the group's, independent of the other individual orderings.
5. *Conflict resolution.* If $M_1 \sqsubset_i M_2$ for some i, then $M_1 \sqsubset M_2$ or $M_2 \sqsubset M_1$. That is, if two interpretations are comparable in an individual order, then they are comparable in the global order.

Technically, these desiderata are a bit more general than Arrow's, as his framework requires the orders to be total orders rather than partial orders. The difference is most apparent in the conflict resolution principle, which for Arrow is implicit in the requirement that the global order be total.

Collective rationality is just a statement of the aggregation framework in preferential nonmonotonic logics. It merely stipulates the goal for aggregation policies, that they define general methods for combining multiple preference criteria.

The Pareto principle is the least assailable of the axioms. Any aggregation function that violated a unanimous preference would be clearly unacceptable.

IIA has been perhaps the most controversial condition among social choice theorists. In the logical context, however, it is precisely the kind of syntax-independence that we would like to enforce. Without IIA, the aggregation function could depend on arbitrary syntactic features such as the introduction of a new predicate symbol.

Dictators are just as unwelcome in preferential nonmonotonic logics as they are in human societies. If we need a sovereign authority there is not much point to the decentralized representation of preferences in the first place, since dictatorial theories presuppose that a unified theory exists, obviating the need for combination of criteria. For example, Konolige's hierarchic autoepistemic theories [Konolige, 1988] represent a dictatorial approach where each node of the hierarchy exercises tyrannical authority over its subordinates. Specifying the hierarchy is tantamount to dictating the resolution of potential conflicts, in effect directly expressing the global preference order. This strict chain of command violates the underlying modularity principle motivating aggregation.

The conflict resolution condition rules out skepticism by mandating that the global order commit one way or the other whenever the individual orders express a preference. Some authors defend skepticism in the face of any conflict. Pollock [1987], for example, argues that belief should be based on epistemically defensible positions. But uniformly skeptical or noncommittal strategies are too weak to serve as universally appropriate theories of default inference. As we saw earlier, the agent cannot always rationally choose to remain skeptical about questions concerning actions that are very important to an agent's prosperity. This is true for the case in which there is too much information (i.e., conflicting preferences) as well as the case in which there is too little information (i.e., incomplete beliefs). In either case, it may be better to adopt a stance on some issue and risk error than to take no stance at all and risk paralysis (see [Doyle, 1989] for examples and further discussion). This does not mean that universal theories should never indicate skepticism—actions presenting enormous risks often call for skepticism—only that universal theories should not uniformly result in skepticism regardless of the conflict.

The following theorem states that the desirable and apparently reasonable properties enumerated above are not simultaneously satisfiable by any aggregation policy for preferences expressed by total orders.

Theorem 1 *No aggregation policy mapping total individual orders to a total global order satisfies the properties 1-4 above.*

Proof: With the restriction to total orders, this theorem is exactly Arrow's theorem. For a proof of the original result see Arrow [1967] or Roberts [1976, Chapter 7].□

4.2 Default logic

To examine the satisfiability of these aggregation principles, let us consider aggregating a set of default rules $P : Q/R$ in the sense of Reiter [1980]. In order to express preferences about when to be skeptical and when

to commit to belief, we employ models which describe belief states as well as the contents of beliefs. For our purposes, we may employ Moore's [1984] models for autoepistemic logic.[2] In this semantics, each model M is a pair $M = (K, V)$ of an ordinary valuation V and a Kripke structure K. A Kripke structure contains a set of possible worlds and an "accessibility" relation between pairs of possible worlds. The truth of a formula is evaluated with respect to each world, and a formula of the form LP (read "P is believed") is true in a world W just in case P is true in every world accessible from W. In Moore's semantics, each K is required to be what he calls a "complete" structure for the modal logic S5, that is, an equivalence relation in which every possible world is accessible from every possible world. Moore proves such models are in exact correspondence with stable autoepistemic theories, that is, deductively closed sets of sentences which contain LP whenever they contain P and which contain $\neg LP$ (read "P is not believed") whenever they do not contain P.

With such interpretations, we may express default rules as preferences in a very natural way. Let us first introduce a bit of helpful notation. If p and q are mutually inconsistent sentences, they are satisfied by disjoint sets of models, and we write $p \prec q$ (q preferred to p) to mean that $M \sqsubset M'$ iff $M \models q$ and $M' \models p$, and that $M \not\sqsubset M'$ for all models M, M' of p and all models M, M' of q. In other words, the models of p (resp. q) are all equally preferable (i.e., the agent is indifferent among them), but all models of q are preferred to all models of p.

We may then express a preference for skepticism about P by

$$LP \vee L\neg P \quad \prec \quad \neg LP \wedge \neg L\neg P,$$

which says that believing neither P nor its negation is preferred to believing either. A preference for credulity about P is expressed by

$$\neg LP \wedge \neg L\neg P \quad \prec \quad LP \vee L\neg P,$$

which is just the opposite of the preference for skepticism. Similarly, a default rule $P : Q/R$ expresses the preferences $\sigma \prec \sigma' \prec \sigma''$ (read transitively), where

$$\begin{aligned} \sigma &= LP \wedge \neg L\neg Q \wedge \neg LR, \\ \sigma' &= LP \wedge L\neg Q, \text{ and} \\ \sigma'' &= \neg LP \vee (LP \wedge \neg L\neg Q \wedge LR). \end{aligned}$$

That is, if P is believed, the default rule prefers believing R to believing $\neg Q$, and prefers believing either $\neg Q$ or R to believing neither. Note that each of these orders is a total order: for any two models, either they are indifferent to each other, or one is preferred to the other. It never happens that two models are incomparable.

While there may be other motivated ways of interpreting default rules as preference orders over states of belief (cf. [Etherington, 1988]), the interpretation above seems natural in a direct reading, and is corroborated by previous results of Doyle [1983b, 1985] which showed that the extensions of default theories are Pareto-optimal choices, that is, correspond to maximal consistent sets of default-rule preferences.[3]

Theorem 2 *No policy for aggregating default rules into a total global order satisfies the properties 2-4 above.*

Proof: We show that restricting the input orderings to those arising from default rules is not sufficient to forestall the impossibility theorem. Consider three sentences

$$\begin{aligned} \sigma_1 &= LP \wedge \neg LQ \wedge \neg L\neg Q \wedge \neg LR \wedge \neg L\neg R \wedge LS \\ \sigma_2 &= LP \wedge L\neg Q \wedge \neg LR \wedge \neg L\neg R \wedge \neg LS \wedge \neg L\neg S \\ \sigma_3 &= LP \wedge \neg LQ \wedge \neg L\neg Q \wedge LR \wedge \neg LS \wedge \neg L\neg S. \end{aligned}$$

As shown in Table 2, there exist default rules to represent every strict preference order among the models satisfying these sentences. Because the individual orders are effectively unrestricted, Theorem 1 applies directly. □

Default	Preferences
$P : Q/R$	$\sigma_1 \prec \sigma_2 \prec \sigma_3$
$P : \neg R/\neg Q$	$\sigma_1 \prec \sigma_3 \prec \sigma_2$
$P : \neg R/S$	$\sigma_2 \prec \sigma_3 \prec \sigma_1$
$P : \neg S/R$	$\sigma_2 \prec \sigma_1 \prec \sigma_3$
$P : Q/S$	$\sigma_3 \prec \sigma_2 \prec \sigma_1$
$P : \neg S/\neg Q$	$\sigma_3 \prec \sigma_1 \prec \sigma_2$

Table 2: Six default rules which result in all possible preference orders over $\sigma_1, \sigma_2, \sigma_3$.

4.3 A general impossibility theorem

The special case of default rules by itself does much damage to hopes for a unified theory of nonmonotonic reasoning since a general theory should cover at least these. But one might still escape this limitation by dropping the restriction that the orders be total, requiring only the partial orders appearing in Shoham's

[2] One can also formalize these preferences using situations to describe belief states, as in Levesque's logic of explicit belief [Levesque, 1984].

[3] Shoham's [1988] treatment of default logic directly defines a global preference order for each default theory. While these orders recast the global interpretations (extensions) of default theories as maximally preferred models, Shoham's treatment does not address the question of why these global orders best represent sets of individual default rules.

framework. The following theorem shows that the impossibility result recurs if we require global orders to be as complete as the individual orders.

Theorem 3 *No aggregation policy for preferential nonmonotonic logics satisfies the properties 1-5 above.*

Proof: The only difference between the multicriteria nonmonotonic logic problem and the classic social choice setup is that $\sqsubset$ and the $\sqsubset_i$ can be partial orders whereas individual and social rankings are taken to be total. Partiality is constrained, however, by the conflict resolution condition's restriction that the global order be at least as complete as the constituent orders. Therefore, any set of interpretations that is totally ordered by some $\sqsubset_i$ is also totally ordered by $\sqsubset$. The impossibility of the special case of aggregation functions mapping sets of total orders to a total order as in Theorem 2, together with the IIA condition, implies impossibility of the generalized problem where the orders may be incomplete.□

We earlier showed that uniformly skeptical aggregation is not always reasonable. Whether the gap between uniform skepticism and the conflict resolution condition is wide enough to allow a way out remains to be seen.

4.4 Paths toward possibility results

The impact of the impossibility result is proportional to the judged importance of conforming to the premise conditions, as well as the degree to which they need be relaxed in order to achieve "possibility." For social choice, the theorem has had great force due to the apparent reasonableness of the conditions and its demonstrated robustness despite countless mathematician-years spent laboriously tweaking axioms. For nonmonotonic logics, the reasonability of the desiderata is more in question, and more study will be needed to demonstrate the robustness of our results. Paralleling the investigations made in social choice, one can identify two primary options for dealing with impossibility. The first and most direct way out is to restrict or expand the specification of preferences and the basic construction of nonmonotonic logics from them. The second approach attempts to find a compromise among the conflicting desiderata, and to analyze the tradeoffs involved in different compromises.

The impossibility result is fundamentally a statement about the relation between the expressive power of a preferential nonmonotonic logic and the difficulty of combining multiple criteria. To accept the aggregation principles yet avoid the implications of Theorem 3, the language for representing preferences needs to be more or less expressive in some way than the framework presented above.

For example, the impossibility result can be circumvented by expanding the language of preferences to include some expression of intensity of preference (in contrast to the merely ordinal information expressible by the $\sqsubset_i$).[4] More specifically, the ordinal specification of individual preferences can be strengthened in two ways. The first is to allow *intercriteria comparisons*, permitting statements of the form "criterion i likes M_1 more than criterion j likes M_2." The second enhancement introduces *intracriteria intensities*, where i's degree of preference for M_1 over M_2 can be compared to its preferences for M_3 over M_4. Taken alone, intercriteria comparison only opens the door a crack, leading to aggregation policies that are almost-but-not-quite dictatorial (in a precise sense described, for example, by Roberts [1980]). And incorporating only intracriterial intensity comparisons helps not at all. Together, however, the two measures induce a fully cardinal description of preferences (i.e., a numeric measure of degree of preference), which leads immediately to satisfactory aggregation functions of the sort recommended by multiattribute utility theory [Keeney and Raiffa, 1976]. Although it solves the impossibility problem, we suspect that designers of nonmonotonic logics will not be eager to require in effect that numeric utility measures be assigned to every interpretation. Numeric representations are typically avoided because they are excessively precise and present an intolerable specification burden on the source of default assertions. To make this approach palatable, one would have to find some qualitative (direct) expression of the available preference information (going beyond purely ordinal comparisons) from which the numerical measures could be automatically constructed. Unfortunately, this sort of global comparative information is just what seems to be lacking in our intuition in many cases, as indicated in Section 3. Nevertheless, it may be possible to learn pragmatically useful numerical measures through experience.

Similarly, limiting expressive power by restricting the form of individual partial orders that are handled by the aggregation policy can lead to acceptable policies operating over the smaller domain. Theorem 3 declares the impossibility of a completely general aggregation policy but admits that aggregation might be achievable in special cases. Social choice theorists have explored this route in depth, but the special cases they consider (such as single-peakedness, a condition that the candidates be orderable according to one global dimension) do not appear to be viable for the multicriteria preference problem. On the other hand, AI researchers may discover aggregable special cases particularly well-suited for nonmonotonic reasoning, whether or not they make sense in the original social choice context.

If we insist on maintaining the ordinality of constituent preferences and the universality of the aggre-

[4] Strictly speaking, this is not an expansion of expressive power because the intensity information would have to be *mandatory* to guarantee satisfactory aggregation.

gation policy, we must consider which of the desiderata may be abandoned or relaxed. At the extreme, we could simply give up on global rationality, permitting $\sqsubset$ to be intransitive or inconsistent and so wreaking havoc on the semantics of the resulting nonmonotonic logic. Fortunately, more moderate courses are available. We could choose to live with a bit of syntax-dependence, some lack of resolve (skepticism), or dictators (benevolent we hope), as the least of a selection of evils. Intelligent compromise on this issue requires a much better understanding of the tradeoffs we face. A determination of how much syntax-dependence, etc. we must endure in return for aggregation will result from a deeper analysis of the sources of impossibility. If we can characterize a subclass of preference profiles that fully account for the pessimistic conclusions of Theorem 3, we can limit our desiderata violations to that class. This step is simply a less drastic version of the suggestion above that we restrict the expressive power of the language to exclude the problematic cases. For example, while we argued above that skepticism as a response to all conflicts is irrational, it would be less objectionable to suspend commitment when the conflict is further classified as one of the peculiarly difficult instances. To justify this approach, however, we need some way of estimating the likelihoods with which different sorts of conflicts appear and the risks and benefits that skepticism and credulity offer in each of these cases. Such information about the reasoning process in which the conflicts arise can then be used to determine the cases in which suspending judgment is rational.

5 Conclusions

Translating questions about nonmonotonic reasoning into the language of rational inference and social choice yields valuable insights into the design of nonmonotonic logics. The translation provides a rational justification for the non-deductive structure of some nonmonotonic logics, and the impossibility results presented above expose previously unarticulated difficulties in the quest toward universal default mechanisms. The problem is not attributable to the use of logical or mathematical formalisms for describing or mechanizing reasoning, nor is it due to limited computational resources for carrying out reasoning. Our results delimit the nature of feasible forms of rationality for agents integrating preferences from multiple sources, independent of their representational structure, computational power, or extent of knowledge.

To address the problem, we must continue to investigate special theories of reasoning and the conditions under which each of these is to be preferred or to be avoided. We expect that further analysis from the social choice perspective will suggest promising solution approaches, both because it provides the vocabulary for expressing concepts related to aggregation policies, and because it allows artificial intelligence studies to draw on a large literature of detailed investigations of social choice questions.

Acknowledgments

We thank Ron Loui, Joseph Schatz, and Yoav Shoham for valuable discussions.

References

[Arrow, 1963] Kenneth J. Arrow. *Social Choice and Individual Values*. Yale University Press, second edition, 1963.

[Arrow, 1967] Kenneth J. Arrow. Values and collective decision-making. In Peter Laslett and W. G. Runciman, editors, *Philosophy, Politics and Society (third series)*, pages 215–232. Basil Blackwell Oxford, 1967.

[Borgida and Imilienski, 1984] Alexander Borgida and Tomasz Imilienski. Decision making in committees—A framework for dealing with inconsistency and non-monotonicity. In *Non-Monotonic Reasoning Workshop*, pages 21–32, New Paltz, NY, 1984. American Association for Artificial Intelligence.

[Doyle, 1983a] Jon Doyle. A society of mind: Multiple perspectives, reasoned assumptions, and virtual copies. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 309–314, 1983.

[Doyle, 1983b] Jon Doyle. Some theories of reasoned assumptions: an essay in rational psychology. Technical Report 83-125, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1983.

[Doyle, 1985] Jon Doyle. Reasoned assumptions and Pareto optimality. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 87–90, 1985.

[Doyle, 1988] Jon Doyle. Artificial intelligence and rational self-government. Technical Report CS-88-124, Carnegie-Mellon University Computer Science Department, 1988.

[Doyle, 1989] Jon Doyle. Constructive belief and rational representation. *Computational Intelligence*, 1989 (to appear).

[Etherington, 1988] David W. Etherington. *Reasoning with Incomplete Information*. Pitman, London, 1988.

[Ginsberg, 1987] Matthew L. Ginsberg. *Readings in Nonmonotonic Reasoning*. Morgan Kaufmann, Los Altos, CA, 1987.

[Halpern and Moses, 1984] Joseph Y. Halpern and Yoram Moses. Towards a theory of knowledge and ignorance: Preliminary report. In *Non-Monotonic Reasoning Workshop*, pages 125–143, New Paltz, NY, 1984. American Association for Artificial Intelligence.

[Hanks and McDermott, 1987] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[Horty *et al.*, 1987] John F. Horty, Richmond H. Thomason, and David S. Touretzky. A skeptical theory of inheritance in nonmonotonic semantic networks. In *Proceedings of the National Conference on Artificial Intelligence*, pages 358–363. American Association for Artificial Intelligence, 1987.

[James, 1897] William James. *The Will to Believe and Other Essays in Popular Philosophy.* Longmans, Green, and Co., New York, 1897.

[Keeney and Raiffa, 1976] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* John Wiley and Sons, New York, 1976.

[Konolige, 1987] Kurt Konolige. On the relation between default theories and autoepistemic logic. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 394–401, 1987.

[Konolige, 1988] Kurt Konolige. Hierarchic autoepistemic theories for nonmonotonic reasoning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 439–443. American Association for Artificial Intelligence, 1988.

[Levesque, 1984] Hector J. Levesque. A logic of implicit and explicit belief. In *Proceedings of the National Conference on Artificial Intelligence*, pages 198–202. American Association for Artificial Intelligence, 1984.

[Lifschitz, 1986] Vladimir Lifschitz. Pointwise circumscription: Preliminary report. In *Proceedings of the National Conference on Artificial Intelligence*, volume 1, pages 406–410, 1986.

[McCarthy, 1980] John McCarthy. Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1):27–38, 1980.

[McDermott and Doyle, 1980] Drew McDermott and Jon Doyle. Non-monotonic logic—I. *Artificial Intelligence*, 13:41–72, 1980.

[Minsky, 1986] Marvin Minsky. *The Society of Mind.* Simon and Schuster, 1986.

[Moore, 1984] Robert C. Moore. Possible world semantics for autoepistemic logic. In *Non-Monotonic Reasoning Workshop*, pages 21–32, New Paltz, NY, 1984. American Association for Artificial Intelligence.

[Moore, 1985] Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94, 1985.

[Nowakowska, 1973] Maria Nowakowska. *Language of Motivation and Language of Actions.* Mouton & Co., The Hague, 1973.

[Pascal, 1962] Blaise Pascal. *Pensées sur la religion et sur quelques autres sujets.* Harvill, London, 1962. translated by M. Turnell, originally published 1662.

[Pollock, 1987] John L. Pollock. Defeasible reasoning. *Cognitive Science*, 11:481–518, 1987.

[Reiter, 1980] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Roberts, 1976] Fred S. Roberts. *Discrete Mathematical Models.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1976.

[Roberts, 1980] Kevin W. S. Roberts. Possibility theorems with interpersonally comparable welfare levels. *Review of Economic Studies*, 47:409–420, 1980.

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

[Thomason, 1986] R. H. Thomason. The context-sensitivity of belief and desire. In Michael P. Georgeff and Amy L. Lansky, editors, *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, pages 341–360. Morgan Kaufmann, 1986.

[Touretzky, 1986] David S. Touretzky. *The Mathematics of Inheritance Systems.* Morgan Kaufman, Los Altos, CA, 1986.

[Touretzky *et al.*, 1987] David S. Touretzky, John F. Horty, and Richmond H. Thomason. A clash of intuitions: The current state of nonmonotonic multiple inheritance systems. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 476–482, 1987.

Situated Control Rules

Mark Drummond
Sterling Federal Systems
AI Research Branch, Mail Stop: 244-17
NASA Ames Research Center
Moffett Field, CA 94035 U.S.A.

Abstract

The plan net formalism presented in this paper is a language for expressing possible actions independent of any particular plan. Plan nets encode non-deterministic control programs which specify the local conditions under which actions may be taken. A two stage analysis synthesizes *situated control rules* which act to reduce this non-deterministic choice such that all remaining alternatives for action necessarily satisfy any given goals. Action execution can precede situated control rule synthesis: situated control rules simply help ensure that actions are first steps on the way to eventual goal satisfaction. Situated control rules *inform* action; they do not *define* it.

1 Introduction

Two assumptions are often made about the nature of plans in AI systems. First, there is the assumption that *plans are programs*; second, the assumption that *plans are totally or partially ordered sets of operators.* These two assumptions have limited the generality and success of planning in AI. This first section explains why this is so.

Consider the assumption that plans are programs. This is limiting, since an agent must be able to act without plans. If plans are programs then without a plan there is no program for an agent to run. An agent's basic behavioural abilities must be *informed* by plans, and not *defined* in terms of plans. This point has been well made by Agre and Chapman (1988). When there is time to plan, the results of planning should help make behaviour more goal-directed. But when situations warrant, action must be taken without advance planning. A plan should be only one of many possible resources to action. Plans must inform action, and should not be solely responsible for defining it.

Consider next the assumption that plans are totally or partially ordered sets of operators. This assumption limits an agent's ability to robustly execute plans of action. There is no sure way to realize the actions that a plan's operators describe. Executing an instruction inside the computer controlling an agent *may or may not* realize the desired action in the environment. The view of "plans as programs" is derived from the notion of "agent as computer". But the analogy is a poor one. When programming a computer we rely on descriptions of machine instruction behaviour that are provided by the machine's manufacturer. If the machine fails to realize these descriptions we should ask for our money back. The computer is *expected* to live up to these descriptive promises. For an agent, the connections between the operators and the actions they describe is uncertain: actions cannot always be performed as specified in a plan. Sometimes, actions simply *fail.*

Since actions fail, plans cannot be expressed as ordering relations on sets of operators. A plan executor that blindly selects and attempts to execute operators in planned sequence will fail too often to be useful. Instead, a plan executor must carry out explicit checks to see whether or not actions have been successfully executed. To allow this, a plan must specify operators in terms of the overall conditions under which their performance is appropriate. STRIPS *triangle tables* (Nilsson, 1984) do this, as do Schoppers' (1987) *universal plans.* Under this general view, a plan is not a sequence, but is instead a specification of reactions to situations.

2 Overview

Plan nets are a language for describing the basic behavioural abilities of an agent. Situated control rules (SCRs) are synthesized through temporal projection and are used to constrain the behaviours produced by plan nets. Each SCR characterizes the performance of possible actions in terms of an agent's current environment. This approach guides plan execution by doing situation-based action indexing and so better handles the problem of action failure. Action can also be taken by the agent without advance planning, simply by executing the plan net. Any planned behaviour produced by the agent is a result of interactions between its plan net and any SCRs that have been synthesized.

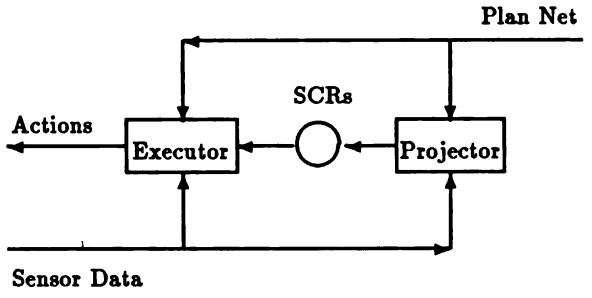


Figure 1: Information Flow

An agent-level sketch is given in figure 1, the components of which interact as follows. The *executor* accepts a plan net and interprets it as a nondeterministic program to be *run*. It also accepts data from sensors describing the state of its environment. Planned behaviour by the executor is the result of SCR input. The *projector* accepts the same plan net as the executor and interprets it as a nondeterministic program to be *analyzed*. The projector analyzes the plan net and produces SCRs to deal with critical choice points in the future. A critical choice point is a situation during execution where it is possible for the executor to take a locally possible action that does not lead to a globally acceptable solution. The executor will always check to see if any SCRs exist that are appropriate to the current situation. If so, the SCRs' advice about what to do next will be heeded. If there are no appropriate SCRs, unplanned execution is still possible. Without reference to the SCR input from the projector, the executor will simply select and attempt to execute any enabled operator in the plan net. The results of such non-deterministic execution are (of course) unpredictable.

The process of SCR synthesis takes time. In some cases, the execution component will act before the projection process is "complete". This simply affects the overall quality of the agent's behaviour, and not its basic ability to execute and project independently. Of course, the hope is that the projector can always remain ahead of the executor. If not then the agent may have to "physically backtrack" to get out of tricky situations (like waiting for the paint to dry after painting itself into a corner).

To focus the discussion more sharply, this paper considers a simple assembly problem that demands a high-degree robustness from an agent. The example is only used to demonstrate plan nets and SCRs. Our approach is more general than this example might suggest. This problem is presented in section 3. Section 4 presents plan nets and SCRs. Section 5 establishes the connection to related work.

3 A Planning and Control Problem

Our example plan projection and execution problem is adapted from Fox and Kempf (1988). In it, we are given a table on which to assemble three blocks in a row: block A on the left, at location 1; block B in the middle, at location 2; and block C on the right, at location 3. The blocks are initially available for placement, and each block can be placed on the table once it's available. The exact means for moving the blocks does not matter: when a block is available it may be placed. The only constraint on assembly is that *block B cannot be placed last*: once A and C are down, there is not enough room to place B. B must be swept in from the left or from the right, and cannot be placed if A and C are in the way. We call this the BNL (B Not Last) problem.

We have to be careful about what it means to "solve" this problem. Suppose that the BNL problem is under the control of a particular assembly agent. We want to define the problem for this agent such that it can reason about the different possible assembly strategies, and select only those strategies where B is not placed last. That is, the agent should be able to reason about possible assembly strategies if it has time to do so, but it should also be able to simply begin the assembly process if there is no time to carry out the required reasoning. Of course if no reasoning is done about the assembly, the agent could easily "deadlock" in the state where blocks A and C are in place; B could not then be placed according to the BNL constraints. The results of reasoning should guide the agent away from this state. Further, for our version of the BNL problem, we want the agent to reason about as many assembly strategies as possible. The motivation for this is *uncertainty*: if the blocks were not available at the problem's start, and if their availability was instead determined by individual block deliveries, then it would be desirable to have the agent consider as many assembly strategies as reasoning time allowed. Relevant strategies would be indexed by the actual block deliveries which occurred during plan execution. While this robustness requirement is not well motivated in our simple version of the BNL problem, it is easy to understand why such robustness is required in general.

It takes three partially ordered plans to represent all assembly strategies for the BNL problem. Using the notation (X, Y) to indicate that action X must occur before action Y, the following three partial orders are needed to give full assembly competence to the BNL executor: $\{(A, C), (B, C)\}$, $\{(B, A), (C, A)\}$, and $\{(B, A), (B, C)\}$. While there is some overlap in the extensions of these partial orders, all three *are* necessary. If only two of the three partial orders are passed to the executor it will be unable to realize all possible assembly strategies.

If one partial order is thought of as one plan then

there is no one plan that can be given to the executor to solve the BNL problem (in the sense discussed above). For instance, suppose that the executor is given $\{(A,C),(B,C)\}$ as its plan. This plan does not specify that block C can be placed first, even though such an action could begin a strategy which achieves the required assembly. There are different possible assembly strategies, and no one partial order can describe all of them. To deal effectively with this problem, we must be able to represent and reason about *disjunction* within a plan.

One might deal with disjunction by defining a plan to be a *set* of partial orders; unfortunately, this turns out to be unsatisfactory from an execution stance. Given the three partial orders, how does the executor index the appropriate planned action? The plan is essentially a disjunction of partial orders. We might expect that the executor will simply perform actions in a sequence compatible with one of the partial orders it has been given. As explained in section 1, sequence following approaches to plan execution will not work in general; instead, actions must be given in terms of the conditions under which their performance is appropriate.

Our approach does not depend on the view of plan execution as sequence following. Instead, the executor *indexes* appropriate operators by current environmental conditions. This approach handles the problem of action failure. If an action is attempted and fails, the environment will reflect the results of the failed action. The state of the environment following the failure can be used to determine the appropriate action to perform next. Of course successful execution is also captured by the indexing approach: when actions are executed as expected, the environment will be in the state predicted by the planner, and the preconditions of the next operator will actually hold. So when actions occur as predicted, the indexing approach generates the same behaviour as the sequence following approach.

One last difficulty lurks within BNL: contingent causal independence. Consider the initial situation for the BNL problem, where all the blocks are available. According to the constraints, blocks A and B could be placed in any order, or in *parallel*. Similarly, blocks B and C could be placed in any order, or in parallel. Alternatively, if B is placed first, then A and C could be placed in any order, or (once again) in parallel. This situation-contingent causal independence makes disjunctive action indexing even more difficult (and interesting!).

4 Plan Nets and SCRs

This section defines and explains plan nets and SCRs. Examples drawn from the BNL problem are scattered throughout. The first few subsections simply define plan nets and the formal machinery needed to do temporal projection.

4.1 Plan Nets

A plan net is a type of Condition/Event system (Reisig, 1985), essentially, a specific sort of Petri Net. A plan net is a bipartite directed graph built from two types of nodes called *conditions* and *operators*. Conditions are facts about the agent's environment that can be true or false (equivalently: can hold or not hold). Operators denote events. The arcs between nodes describe relationships of causation and enablement: operators cause the holding of conditions and the holding of conditions enables the application of operators.

For the rest of this paper, let L_c be a set of terms of the form $f(x_1, x_2, \ldots, x_n)$, where f is an n-ary predicate and each x_i is an argument of f; each x_i may be a variable or constant. Conditions in plan nets will be drawn from L_c; thus, each term in L_c will be interpreted as a fact about the agent's domain which can be true or false. Similarly, let L_o be a set of operator terms. Each operator in a plan net will be drawn from L_o. L_c and L_o must be disjoint: $L_c \cap L_o = \{\}$. An arbitrary finite subset $S \subset L_c$ is called a *case*; S describes a set of domain states, precisely those states in which each condition in S is true. (The symbol "S" is used to suggest the word "state".) Three final bits of notation: R^* denotes the transitive closure of the relation R; $\Pi(O)$ denotes the powerset of the set O; and $\setminus$ denotes set subtraction.

Definition 1 *A* plan net *is a triple* $N = (C, O, F)$, *with* $C \subset L_c$, $O \subset L_o$, *and* $F = (C \times O) \cup (O \times C)$; *both* C *and* O *must be finite.*

So each $c \in C$ is a condition which can be true or false in the agent's domain, and each $o \in O$ is an operator which denotes an event. F is composed of two relations: $(C \times O)$, interpreted as *enables*; and $(O \times C)$ interpreted as *causes*.

Definition 2 *Let* $N = (C, O, F)$ *be a plan net. For each* $o \in O$: $pre(o) = \{c \mid (c, o) \in F\}$ *and* $post(o) = \{c \mid (o, c) \in F\}$. *Elements of* $pre(o)$ *are called the* preconditions *of* o *and elements of* $post(o)$ *are called the* postconditions *of* o. *The functions pre and post are extended to handle sets of operators: for* $P \subseteq O$, $pre(P) = \bigcup_{p \in P} pre(p)$; *similarly for* $post(P)$.

A plan net for the BNL problem is shown in figure 2. Operators are drawn as squares, conditions are drawn as circles, the enablement relation is drawn as arrows from circles to squares, and the causation relation is drawn as arrows from squares to circles. This plan net has four operators. $place(A, 1)$ is an operator which denotes the event of placing block A at location 1. The preconditions for this event are $available(A)$ and $free(1)$. The first precondition is true when block A is available for placement, and the second is true when location 1 is unencumbered. Similar to the operator $place(A, 1)$, we have $place(C, 3)$ at the bottom of the figure. Its preconditions are that C is available, and

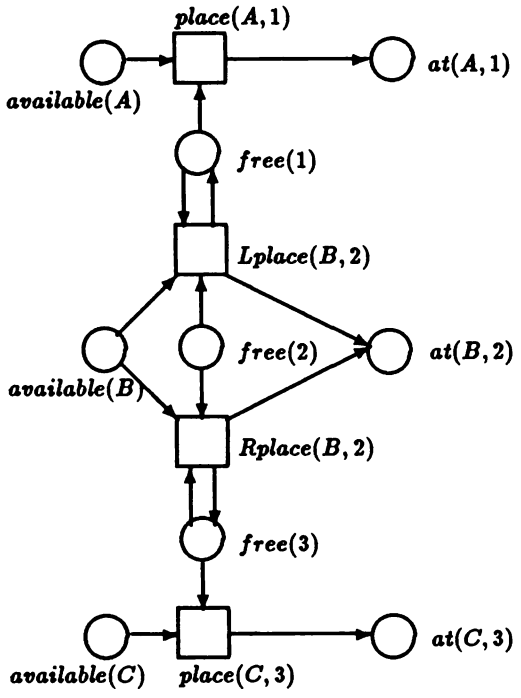


Figure 2: The Plan Net for BNL

that location 3 is free. The postcondition of placing A at 1 is that $at(A,1)$ will be true; similarly $at(C,3)$ is made true by $place(C,3)$.

Block B is treated differently from A and C. There are two ways that B can be placed at location 2: it can be swept in from the left or swept in from the right. The operator $Lplace(B,2)$ denotes the action of sweeping B in from the left, through the space occupied (eventually) by block A. This is why location 1 must be free for the left placement of B. The predicate $free(1)$ is also a postcondition of $Lplace(B,2)$ since the condition is reachieved after the action is finished. The right placement of B, denoted by the operator $Rplace(B,2)$ is similar. It requires that location 3 be free, in order to sweep B through the location to be eventually occupied by C. Either method of placing B results in the condition $at(B,2)$ being true.

4.2 Operator Enablement and Application

To reason about the future we must be able to do temporal projection. Projections are built up from the repeated application of individual operators. So to define a projection, we must first define the conditions under which a single operator may be applied; *i.e.*, when the operator is enabled.

Definition 3 *Let $N = (C, O, F)$ be a plan net, let $S \subset L_c$ be a case, and let $o \in O$ be an operator in the plan net. We say that o is* enabled *in S iff $pre(o) \subseteq S$.*

There are various ways to define operator application. This paper takes an *extremely simple approach*: successor cases are derived by deleting all operator preconditions and adding all operator postconditions. This approach gives us a special sort of STRIPS operator (Fikes & Nilsson, 1971). In STRIPS terminology, our operator's precondition formula would be a conjunction of atomic formulae, and its delete list would contain each of the precondition's conjuncts. There are more general ways to define operator application (Drummond, 1986; Ginsberg & Smith, 1987a, 1987b), but this simple approach will suffice for current purposes. The results of this paper do not depend on any particular definition of operator application.

Definition 4 *Let $N = (C, O, F)$ be a plan net, let $S \subset L_c$ be a case, let $o \in O$ be an operator in the plan net, and let o be enabled in S. The* successor *to S under o is $S' = (S \setminus pre(o)) + post(o)$. This is written as $S \stackrel{o}{\Rightarrow} S'$.*

4.3 Causal Independence

Events can occur in parallel when they are causally independent. Of course there are other situations where events *must* occur in parallel; causal independence simply *allows* parallelism. Temporal projection must be able to use causal independence when reasoning about parallelism.

Definition 5 *Let $N = (C, O, F)$ be a plan net, let $S \subset L_c$ be a case, and let $o_1, o_2 \in O$ be operators in the plan net.*

1. *Operators o_1 and o_2 are* causally independent *iff $pre(o_1) \cap pre(o_2) = \{\}$.*
2. *A subset of the plan net's operators $P \subseteq O$ is* free from interference *in S iff $\forall o \in P$, o is enabled in S and $\forall o_1, o_2 \in P$ such that $o_1 \neq o_2$, o_1 and o_2 are causally independent.*
3. *Let $P \subseteq O$ be a subset of the operators in N that are free from interference in S: the* successor *to S under P is $S' = (S \setminus pre(P)) + post(P)$, written as $S \stackrel{P}{\Rightarrow} S'$.*

The definition of causal independence must vary with the expressiveness of the operator language. Since our operator application mechanism assumes that all preconditions are deleted, the definition of causal independence can exploit locality of possible change: all that is required for two operators to be independent is that their preconditions be disjoint. Other operator languages require different definitions of causal independence. For many languages, the definition does not get very complex. For instance, it is easy to give an appropriate definition for STRIPS operators, the AI planning industry standard.

4.4 Plan Net Projection

Extending the definition of operator application to cover sets of operators affects the structure of the projection. Typically, state-space structures are defined such that state-transitions describe the application of single actions. Not so here. With an application mechanism defined on sets of operators, transitions in a projection can no longer be labelled with individual operators; instead, transitions must be labelled with sets of operators. Operators in a set labelling a transition will have been proven to be free from interference in the case that anchors the transition.

Definition 6 *A tuple $G = (D, A)$ is called an (*arc labelled oriented*)* graph *over L iff D and L are sets such that $A \subseteq (D \times L \times D)$. The elements of D, L, and A are called* nodes, arc labels, *and* arcs *respectively.*

Definition 7 *Let $G = (D, A)$ be a graph over L. For $i = 1, 2, \ldots$, let $p_i = (d_i, l_i, d_i')$. $w = p_1, p_2, \ldots$ is called a* path *in G iff for $i = 1, 2, \ldots, d_i' = d_{i+1}$. Paths may also be written as $w = d_1 l_1 d_2 l_2 \ldots$*

We can now define the projection of a plan net N from an initial case (state characterization) S. The projection is simply a graph: the nodes are cases and characterize future possible domain states; the arc labels are sets of operators and characterize the simultaneous application of domain actions. Consistent with this, a path in a projection graph will be interpreted as a *behaviour*.

Definition 8 *Let $N = (C, O, F)$ be a plan net, and let $S \subset L_c$ be a case. The* projection *of N from S is the arc labelled oriented graph $G = (D, A)$, where the set of nodes, D, is given by $D = \{S' \subset L_c \mid (S, S') \in T^*\}$. The relation T is given by $T = \{(S_1, S_2) \subset (L_c \times L_c) \mid \exists P \subseteq O : S_1 \stackrel{P}{\Rightarrow} S_2\}$. The set of arcs, A, is given by $A = \{(S_1, P, S_2) \subset (L_c \times L_o \times L_c) \mid S_1, S_2 \in D \wedge S_1 \stackrel{P}{\Rightarrow} S_2\}$.*

This approach allows temporal projection to exploit operator independence when possible. Assume that the projection process has arrived at a case (future state characterization) where there are n different operators that are enabled. The cost of proving that this set is free from interference is $O(n^2)$ – simply the cost of performing a pair-wise causal independence test. If operators cannot be proven to be causally independent projection can still proceed. Of course the analysis cost then approaches $O(n!)$, since it might be necessary to reason about all possible orders of operator application. But in the best case, when all enabled operators are free from interference, the cost of analysis goes down dramatically. Contrast this with the classical approach to reasoning about operator application (Chapman, 1985), where pair-wise causal independence is *assumed*. If the assumption proves false then the reasoning mechanism simply fails. With our approach, the operators are *assumed to conflict* with each other, unless the proof of causal independence goes through. Only when operators can be proven to be causally independent can they be applied in parallel.

```
project(N,S,X,G) {
  if X > 0 then {
    A = free_from_interference(S,N);
    for P in A {
      S1 = apply(P,S);
      if S1 is not in G then {
        G = add_case(S1,G);
        G = add_labelled_arc(S,S1,P,G);
        project(N,S1,X-1,G);
      }
      else G = add_labelled_arc(S,S1,P,G);
    }
  }
  return(G);
}
```

Figure 3: The Plan Net Projection Algorithm

A projection algorithm is given in figure 3. The algorithm accepts a plan net `N`, an initial case `S`, a depth cutoff indicator `X`, and an initial projection graph `G` (which contains `S`) as arguments; it returns the projection of `N` from `S` to an event horizon `X` events into the future. The depth cutoff argument allows the process to terminate before an exhaustive projection has been formed. The case `S` is assumed to be in the graph `G` to begin with – initially, `S` will be *all* that is in `G`. The function `apply(P,S)` accepts a set of operators, `P`, and a case, `S`; it returns the successor to `S` under the application of `P`. This function implements $S \stackrel{P}{\Rightarrow} S'$. The function `free_from_interference(S,N)` accepts a case, `S`, and a plan net, `N`; it returns a set of sets of operators in `N`, each of which has been proven to be free from interference in `S`. The function `add_case(S,G)` accepts a case, `S`, and a projection `G`; it returns the projection which results from adding `S` to `G`. The function `add_labelled_arc(S1,S2,P,G)` accepts two cases, `S1` and `S2`, a set of operators, `P`, and a projection `G`; it returns the projection which results from adding an arc in `G` from `S1` to `S2`, labelled with `P`.

This algorithm is guaranteed to terminate, and not simply due to the horizon cutoff argument, `X`. Suppose that `X` is set to ∞: the algorithm will still terminate. The cases that a plan net can produce are determined by its conditions, and each condition can either hold or not hold. Each plan net has a finite number, n, of conditions. Thus, the maximum number of cases that a plan net can give rise to is 2^n. So while the algorithm is exponential in complexity, it *is* guaranteed to terminate, given an arbitrary value for `X`.

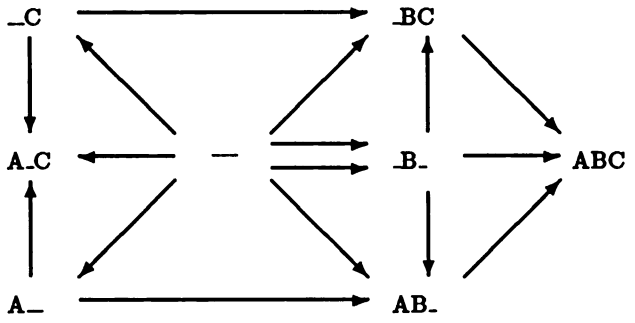


Figure 4: The Basic Structure of the BNL Projection

There are various ways to consider improving the algorithm's efficiency. The projection algorithm is performing a chronologically organized search through a space of world state descriptions. One way to restrict this search is through the use of an operator relevancy analysis such as Means-Ends Analysis (MEA). MEA can be used as a heuristic to control the number of alternatives considered at each point in the search. Like any search procedure, projection requires heuristic knowledge to make it efficient. We return to this topic in section 5.

The basic structure of the complete BNL plan net projection from the problem's initial case is given in figure 4. The initial case is drawn as "___". All blocks are available for placement, but this is not shown. In other cases, blocks are located as indicated by the appropriate letter. So "A__" indicates the case where only block A has been placed; "AB_" indicates the case where blocks A and B have been placed; *etc.* Possible transitions between cases are indicated by arcs in the figure. For clarity, these arcs are not labelled. It should be obvious what the actual projection cases and arc labels are, given the plan net of figure 2.

Seven transitions are possible from the initial case. The placement of A and the right placement of B are causally independent, so there is an arc from "___" to "AB_". Similarly, block C and the left placement of B are independent, leading to "_BC". Two of the transitions from the initial case deal exclusively with the individual placement of block B: it may be swept in from the left or from the right; from the initial case, both means of placing B are enabled. The placements of A and C are also causally independent in the initial case. Notice however that placing A and C first leads to the case "A_C", from which no further progress is possible. It is this case that makes the BNL problem interesting (to the extent that it is). Not all projection paths arrive at the final case "ABC", so some care is required during execution. The projection indicates that there are some executions which achieve the goal, and that there are some executions which do not. This is not unreasonable: the plan net simply specifies what *can* happen, not what *should* happen.

4.5 A Goal Language

Our goal language is based on the work of Ben-Ari *et al* (1981). Goals are expressions which post constraints on acceptable projection paths. The agent must ensure that goals are *necessarily* satisfied by a given set of situated control rules. This means that the agent must have access to modal operators defining the notions of necessity and possibility. The language L_g contains all of the formulae needed to deal with these notions. L_g is recursively defined as follows.

Definition 9 *All elements of L_c are in L_g. If p and q are in L_g then $\neg p$, $p \wedge q$, $p \vee q$, $\Box maint(p)$, $\Box ach(p)$, $\Box obt(p)$, $\Diamond maint(p)$, $\Diamond ach(p)$, and $\Diamond obt(p)$ are in L_g. Nothing else is in L_g. Elements of L_g that do not contain $\Box$ or $\Diamond$ are called* goals*; goals in L_g that do not contain maint, ach or obt are called* simple *goals.*

The satisfaction of an L_g wff is defined with respect to a specific case in a given plan net projection. Consistent with standard interpretations, $\Box maint(p)$ is true in a projection case S if and only if p is necessarily maintained in all projection paths rooted at S; similarly, $\Diamond ach(p)$ is true in S if and only if p is possibly achieved in some path rooted at S. This is expressed more precisely by the following definition.

Definition 10 *Let $N = (C, O, F)$ be a plan net, let $S \subset L_c$ be a case, let $G = (D, A)$ be the projection of N from S, let $p, q \in L_g$, and let W be the set of paths in G rooted at S, given by $W = \{w = d_1 l_1 d_2 l_2 \ldots \mid d_1 = S\}$. An ordering relation $>$ on cases in projection paths is defined by $d_i > d_j$ iff $i > j$.*

1. *For $p \in L_c$, $S \models p$ iff $p \in S$.*
2. *$S \models \neg p$ iff $S \not\models p$.*
3. *$S \models p \wedge q$ iff $S \models p$ and $S \models q$.*
4. *$S \models p \vee q$ iff $S \models p$ or $S \models q$.*
5. *$S \models \Box maint(p)$ iff $\forall w \in W\ \forall d \in w : d \models p$.*
6. *$S \models \Box ach(p)$ iff $\forall w \in W\ \exists d \in w : d \models p$.*
7. *$S \models \Box obt(p)$ iff*
   $\forall w \in W\ \exists d_i \in w : d_i \models p \wedge \forall d_j > d_i : d_j \models p$.
8. *$S \models \Diamond maint(p)$ iff $\exists w \in W\ \forall d \in w : d \models p$.*
9. *$S \models \Diamond ach(p)$ iff $\exists w \in W\ \exists d \in w : d \models p$.*
10. *$S \models \Diamond obt(p)$ iff*
    $\exists w \in W\ \exists d_i \in w : d_i \models p \wedge \forall d_j > d_i : d_j \models p$.

The goal for BNL in L_g is

$$ach(at(A, 1) \wedge at(B, 2) \wedge at(C, 3)).$$

We will assume that all goals given to the system are goals of *necessity*; *i.e.*, if the goal given to the system is p then the system attempts to ensure that "$WM \models \Box p$" from the agent's current case, WM (for World Model), in terms of its current plan net and projection.

When a formula p is evaluated with respect to a specific case S in a projection, there are three possible outcomes: p might be necessarily true ($S \models \Box p$), possibly true ($S \models \Diamond p$ or $S \models \Diamond \neg p$), or necessarily false ($S \models \Box \neg p$). Each of these situations demands a different response from the agent.

A goal will be *necessarily true* in a projection case if it is true for all paths that are rooted in that case. If a goal is necessarily true then no extra work is required: all possible behaviours already satisfy the goal.

A goal will be *possibly true* in a projection case if it is true for some of the paths rooted in the case and false for some others. Possible truth indicates that some extra work is required of the agent. Since not all behaviours satisfy the goal, only those that do must be realized.

A goal p will be *necessarily false* in a projection case S if there are no paths rooted in S which make p true. In such a situation, there is nothing the agent can do to make the goal true: the goal cannot be satisfied, no matter what behaviours are produced.

If the projection does not include at least *one* path that satisfies the required goal then there is no way of synthesizing a satisfactory plan. On the other hand, if *all* paths through the projection satisfy the goal then no plan (and no planning) is necessary: all possible behaviours are acceptable. Such luck is unlikely in practice. More typically, there will be *some* behaviours that achieve the goals and some others that do not. The agent must transform this *possibility* into *necessity*.

4.6 Situated Control Rules

The goal for BNL is only possibly true in the initial projection case (figure 4). In total, there are three cases in the projection in which the goal is only possibly true. We call such cases *critical choice points*. The critical choice points in the BNL projection are, graphically: "__", "A__" and "__C". There is only one case in which the goal is necessarily false: "A_C". The goal is necessarily true in all other projection cases.

The problem is that the executor does not have access to global information in the plan net projection. It must make local decisions about what it is *best* to do next, based only on information regarding what it is *possible* to do next. The executor cannot be expected to recognize critical choice points simply by sensing the state of its environment. Information must be communicated to the executor regarding behaviours that satisfy goals. This information must be structured in a way that informs the executor about what to do in terms of directly-sensible environmental conditions. The information cannot be in the form of operator sequences. Objections to the sequence-following approach were raised in section 1.

We give advice to the executor in the form of *Situated Control Rules* (SCRs). The antecedent of an SCR is a well-formed-formula that can be evaluated in the agent's current domain state. If the antecedent of an SCR is satisfied then the rule is applicable. The consequent of an SCR is a set of sets of operators: any one of the operator sets can be executed. Execution of the operators is guaranteed to lead to the satisfaction of the agent's goal. Each operator set in the consequent of an SCR must be free from interference in the projection case in which the SCR's antecedent holds.

Definition 11 *Let $N = (C, O, F)$ be a plan net, let $S \subset L_c$ be a case, let $G = (D, A)$ be the projection of N from S, and let $p \in L_g$ be a goal.*

1. *A* situated control rule *(or SCR) is a rule of the form $I \rightarrow E$, where $I \in L_g$ is a simple goal and $E \subseteq \Pi(O)$.*
2. *An SCR $I \rightarrow E$ is* sound *in S with respect to p iff*
   - *$S \models I$ and*
   - *$S \models \Diamond p$ and*
   - *$\forall e \in E$: e is free from interference in S and*
   - *$\exists S' \subset L_c$: $(S, e, S') \in A$ and $S' \models \Diamond p$.*

An SCR provides local information about what the executor can do *next*. The notion of SCR *soundness* is critical in this context. An SCR is sound in a case with respect to a goal only if it is applicable to the case, if the goal is already possible from the case, if each of the operator sets suggested by the SCR is free from interference, and if the application of each of the suggested operator sets gives rise to a case from which the goal is still possible. If sound SCRs exist for every critical choice point, the executor will never take an incorrect action. Sound SCRs guarantee that local execution choices always lead to possible goal achievement.

This last point brings up the problem of SCR coverage: does the set of available SCRs completely cover every possible critical choice point? SCR coverage is necessarily occasionally incomplete, since it will often be infeasible to compute a complete set of SCRs for a problem due to resource constraints imposed on the projection process. If the entire projection *is* computed then appropriate SCRs can be formed to deal with *all* critical choice points. But if the projection is only partially computed, SCRs coverage might be incomplete. Incomplete coverage might result in the synthesis of unsound SCRs. This is a problem for any search process with a limited event horizon: if search is not carried out completely, surprises can hide beyond the event horizon considered.

As an example, let's consider the SCRs for BNL. There are three critical choice points in the complete BNL projection; thus, we need three SCRs. Recall the critical choice points that must be covered: "__", "A__" and "__C". The rules have been placed in figure 5 for easy reference.

The first SCR covers the case "__". It indicates that when all the blocks are available, A may be placed by

$$available(A) \wedge available(B) \wedge available(C) \wedge free(1) \wedge free(2) \wedge free(3) \rightarrow \{\{place(A,1)\}, \{Lplace(B,2)\}, \{Rplace(B,2)\}, \{place(C,3)\}, \{place(A,1), Rplace(B,2)\}, \{Lplace(B,2), place(C,3)\}\}$$

The First SCR

$$at(A,1) \wedge available(B) \wedge available(C) \wedge free(2) \wedge free(3) \rightarrow \{\{Rplace(B,2)\}\}$$

The Second SCR

$$available(A) \wedge available(B) \wedge at(C,3) \wedge free(2) \wedge free(1) \rightarrow \{\{Lplace(B,2)\}\}$$

The Third SCR

Figure 5: The BNL Situated Control Rules

itself; that C may be placed by itself; that B may be swept in from the left or from the right; that A and the right placement of B are free from interference and may be executed in parallel; and that C and the left placement of B are free from interference, and may also be executed in parallel.

In the first SCR, one might think that the set $\{place(A,1), Rplace(B,2)\}$ entails that $\{place(A,1)\}$ is also acceptable by itself. This is not always true. If the projection has only considered maximally parallel operator applications, then cases which result from arbitrary event interleavings might not have been constructed in the projection. For our example, these intermediate cases *have* been considered, since a complete projection has been constructed. This is communicated to the executor by explicitly listing the individual actions that may be performed.

The second SCR covers the case "$A\_$". It permits placing B only through a right sweep. The other possibility for immediate execution from this case is $place(C,3)$. The SCR does not permit this, since placing C gives rise to a case in which the goal is necessarily false.

The third SCR covers the case "$\_C$". Symmetric with the second SCR, it precludes the placement of A when such action would lead to necessary failure. In the case covered by this rule, the only safe action to perform is $Lplace(B,2)$; that is, B must be swept in from the left.

4.7 SCR Synthesis

An SCR synthesis algorithm is given in figure 6. This algorithm accepts three arguments: `G`, a plan net projection; `S`, a particular case in `G`; and `W`, a set of paths in `G` rooted at `S`. It returns a situated control rule for the case `G` which permits only the execution of operators which comprise the first transition in any of the paths in `W`.

```
synthesize(G,S,W) {
  SCRs = nil;
  for w in W {
    Ante = nil;
    for i = length(w) to 2 {
      Ante = Ante + pre(arc_label(i,w));
      Added = case(i,w) - case(i-1,w);
      Ante = Ante - Added;
    }
    Ante = Ante + pre(arc_label(1,w));
    SCRs = make_scr(Ante,arc_label(1,w));
  }
  return(simplify(SCRs));
}
```

Figure 6: The SCR Synthesis Algorithm

SCR formation for a case and path is, in principal, quite trivial: the antecedent of the SCR is simply the conjunction of all conditions in the case; the consequent of the SCR is simply a set containing the operator set labelling the first arc on the path. But in practice, some discretion is required in forming the SCR's antecedent. Not all conditions in a case are relevant to the success of the given path, and putting all conditions in the antecedent will doom the executor to checking endless irrelevant details. To find just those conditions that *are* relevant, we must scan along the given path, forming a conjunction of conditions relevant to the operators encountered during the scan. This is exactly what the algorithm in figure 6 does. Of course, in a problem as simple as BNL, the two approaches produce the same result, since there are no conditions given in the initial case that are irrelevant to the operators in the plan net being projected.

The algorithm depends on the following primitive functions. The function `length(w)` accepts `w`, a path; it returns an integer indicating the length of `w`. The function `arc_label(i,w)` accepts an integer `i` and a path `w`; it returns the `i`th arc label in `w`. The function `case(i,w)` accepts an integer `i` and a path `w`; it returns the `i`th case in `w`. The function `pre(O)` accepts a set of operators `O`; it returns the set of preconditions of these operators. The function `make_scr(C,O)` accepts a set of conditions `C`, and a set of operators, `O`; it returns an SCR of the form $I \rightarrow E$, where I is the conjunction of the conditions in `C` and E is a set containing the set of operators in `O`. The function `simplify(S)` can do a variety of things, depending on how it is implemented. For now, it simply accepts a set of SCRs, `S`; it returns another set of SCRs derived from `S` by packing together SCRs in `S` with identical preconditions. This is how the SCRs for the BNL problem end up with mul-

tiple operator sets in their consequents: many of the operator sets have identical preconditions, and this lets `simplify` package them together into a single SCR.

The synthesis algorithm will always terminate, provided that the set of paths, `W`, is finite, and provided that each path in `W` is also of finite length. The first of these requirements is guaranteed by the projection algorithm: projection can only produce finitely many states, and so must terminate eventually. If paths are formed by including each state at most once, then all paths will also be of finite length. The complexity of the synthesis algorithm is polynomial in the number of projection cases in `W`.

One might wish to guarantee that the synthesis algorithm always generates sound SCRs. Attempts at providing such guarantees are, however, misguided. The synthesis algorithm is given a projection, a case, and a set of paths rooted in that case. It generates SCRs that simply index the first operator set in each path by relevant conditions in the case. So if the first operator set in each path is sound, then the SCRs produced by the synthesis algorithm will also be sound. SCR soundness is determined by the path selection mechanism, and paths are selected on the basis of goal achievement. So if paths given to the synthesis algorithm necessarily satisfy the agent's goals, the SCRs it returns will do this as well.

4.8 Execution

A plan net is a non-deterministic program for the executor to run. We adopt the approach taken in the STRIPS Planex system (Nilsson, 1984), where an *Intermediate Level Action* (ILA) is given for each operator. The execution of an operator becomes a call to the corresponding ILA. The ILA implements what the operator describes.

A simple algorithm for an executor is given in figure 7. The algorithm accepts two arguments: `N`, a plan net to run, and `R`, a set of situated control rules. This executor simply checks to see if there are any applicable SCRs in the given set: if there are, it acts on the advice of one of them, randomly selected. If there are no applicable SCRs then a random selection is made from among the operators in the plan net that are enabled in the current case ("world model", or `WM`). If the SCRs are sound and have complete coverage of the problem's state space, this executor will never take an incorrect action.

The translation from operator to ILA is carried out by the routine `exec(P)`; it accepts a set of operators, `P`, and calls the ILAs that implement each of the actions described by the operators in `P`. The function `app_in(S,R)` accepts a case describing the current state of the environment, `S`, and a set of situated control rules, `R`; it returns a subset of `R`, the antecedents of which are satisfied in `S`. The function `choose_SCR(R)` accepts a set of SCRs, `R`; it returns one

```
execute(N,R) {
  AppSCRs = app_in(WM,R);
  ToDo = choose_SCR(AppSCRs);
  Operators = choose_OPS(consequent(ToDo));
  if Operators = nil then {
    Poss = free_from_interference(WM,N);
    Operators = choose_OPS(Poss);
  }
  exec(Operators);
}
```

Figure 7: The Plan Net Execution Algorithm

SCR randomly selected from this set. The function `consequent(T)` accepts an SCR, `T`; it returns the consequent part of this SCR. The function `choose_OPS(P)` accepts a set of operator sets, `P`; it randomly selects and returns one operator set from `P`. The function `free_from_interference(S,N)` is the same as used in the algorithm for plan net projection: it accepts a case, `S`, and a plan net, `N`; it returns a set of sets of operators in `N`, each of which has been proven to be free from interference in `S`. The case `WM` is assumed to be globally available throughout the executor, and describes the state of the agent's environment during execution.

5 Related Work

The current paradigm for plan generation begins with NOAH (Sacerdoti, 1975) and NonLin (Tate, 1977); this approach has been extended by others (Vere, 1981; Wilkins, 1984, Currie & Tate, 1985), but the core idea remains the same. A planner searches through a space of incomplete plans, each of which is partially ordered. Each plan is incomplete in the sense that it must be further refined to be considered acceptable. Transitions through the space of partial plans correspond to plan refinements; typically, operator introductions, variable bindings and additional operator orderings. The choices in this search space have very little to do with choices open to the executor regarding possible actions to perform.

We have defined the search space to be a state-space structure in which the nodes describe domain states and the transitions describe the occurrence of possible domain actions. This search space is *chronologically organized* in the sense that choice in the search space corresponds to, and subsequently directs, the choice of *next relevant domain action*. The advantage of our search space is that choices in the search correspond directly with choices of which action to perform in a situation; thus, incremental search necessarily incrementally informs execution. This is not so with a classical planner. The disadvantage of our approach is that there is not yet any provision for goal directedness in the projection process. Indeed, this is why

we have avoided the term "planning", and used the less dangerous phrase "temporal projection" instead. Some sort of means-ends analysis mechanism will be integrated into the projection process in the near future.

Suchman's (1987) ideas on *situated action* have had significant impact on research within AI. Her study of action-in-context has provided insight on the relationship between planned and unplanned activity. Schoppers' (1987) work on *universal plans* has motivated parts of our approach. His view of planning as the synthesis of reactions to situations has has affected our work, but there are significant differences. Universal plans are typically seen as a compiled response to *all* possible situations. SCRs do not need to cover all possible situations. When SCRs exist, they can be used to guarantee goal satisfaction. When they do not exist, action can still be taken. In contrast, universal plans seem to be the only mechanism available for generating action. If there is no universal plan, then nothing can be done. Universal plan formation is also a one-shot process: the planner must produce a complete universal plan before the executor is called. In contrast, plan nets allow incremental SCR synthesis.

STRIPS *triangle tables* (Nilsson, 1984) are a representation for plans that specify reactions to situations. Triangle tables can give reactions to any one of a set of situations, provided that the situations occur in a pre-specified total order. That is, triangle tables determine a total order on actions to be executed, and cover action failure (where actions must be repeated) and serendipitous goal achievement (where actions can be skipped). Our work can be viewed as an generalization of the triangle tables idea. Triangle tables cannot represent disjunctive behaviours, while SCRs can. Triangle tables also cannot deal with causally independent actions, and this is a major motivation in the theory of SCRs.

Recent work by Nilsson (1988) considers the production of behaviour through *action networks*. It appears that these action networks are more expressive than plan nets. While no techniques for action network synthesis have been presented yet, we expect to be able to borrow from some of Nilsson's work on action net abstraction.

Rosenschein and Kaelbling's (1987) work on the theory of *situated automata* is in part an attempt to provide a theory of high-level symbolic control. They view a machine as situated in an environment, and provide a formal characterization of the machine's knowledge in terms of objective correlations between states of the machine and states of that environment. The GAPPS system (Kaelbling, 1988) is part of this work. GAPPS is a rule compiler: it accepts symbolic goal-reduction rules and produces a circuit which realizes the behaviour inherent in the rules. Each individual rule is expressed in terms of a locally defined *leads-to* operator. The problem of goal interactions between rules must be sorted out by the programmer. GAPPS assumes that the rules' recommendation for action can be composed conjunctively by merging the individual components of each recommendation. It is the business of classical planning to sort out such interactions. Our work deals with goal interactions by doing temporal projection: futures that do not satisfy all goals are prevented from occurring through the creation of appropriate SCRs.

Minton's (1988) work on the Prodigy system is similar in some respects to ours. Prodigy is a general-purpose planner that can improve its performance over time. It improves by learning knowledge regarding correct and incorrect decisions made during the planning process. This knowledge is expressed in *search control rules*. Search control rules are used during subsequent planning to improve the decisions made regarding goal ordering, variable binding and operator introduction. Although Prodigy's search control rules are learned and used across different problem instances, they are similar to our SCRs. Search control rules and situated control rules both express local choice information in terms of global objectives. However, Prodigy defines its search space differently. We use a chronologically organized space, where decisions in the search correspond to actions to perform in the domain. Prodigy defines its search to be through a space of partial plans. Thus, choices in its search correspond to goal orderings, variable bindings, and operator introductions. Future work will examine the relationship with Prodigy's search control rules more closely.

Acknowledgements

Mike Uschold, John Bresina and Dave Thompson provided many useful comments on this paper at extremely short notice. Ken Currie and I developed some of these ideas together when I worked at the AI Applications Institute in Edinburgh. Discussions with Stan Rosenschein, Leslie Kaelbling, Steve Minton and Monte Zweben have helped me clarify my thoughts on many issues and occasions.

References

[Agre and Chapman, 1987] Phil Agre and David Chapman. Pengi: An Implementation of a Theory of Activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268–272, Seattle, Washington, July 1987. American Association for Artificial Intelligence.

[Agre and Chapman, 1988] Phil Agre and David Chapman. What are Plans For? Artificial Intelligence Laboratory Memo 1050, Massachusetts Institute of Technology, 1988.

[Ben-Ari, Manna and Pnueli, 1981] Mordechai Ben-Ari, Zohar Manna, and Amir Pnueli. The Temporal

Logic of Branching Time. 8th ACM Symposium on Principles of Programming Languages, pages 164–176, 1981.

[Chapman, 1985] David Chapman. Nonlinear planning: a rigorous reconstruction. In *Proceedings of the Ninth Joint Conference on Artificial Intelligence*, pages 1022-1024, Los Angeles, CA, 1985. International Joint Committee on Artificial Intelligence.

[Currie and Tate, 1985] Ken Currie and Austin Tate. O-Plan: Control in the open planning architecture. In *Proceedings of the British Computer Society Expert Systems '85*, Warwick, U.K., pages 225–240, 1985. Cambridge University Press.

[Drummond, 1986] Mark Drummond. A representation of action and belief for automatic planning systems. In *Proceedings of Reasoning About Actions and Plans*, pages 189–211, Timberline, Oregon, 1986. Morgan Kauffman.

[Fikes and Nilsson, 1971] Richard Fikes and Nils Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence Journal*, Volume 2, pages 189-208, 1971.

[Fox and Kempf, 1988] Barry Fox and Karl Kempf. Planning, Scheduling, and Uncertainty in the Sequence of Future Events. *Uncertainty in Artificial Intelligence*, Volume 2, J.F. Lemmer and L.N. Kanal (editors), Amsterdam, 1988. North-Holland.

[Ginsberg and Smith, 1987a] Matthew Ginsberg and David Smith. Reasoning about action I: A possible worlds approach. Stanford Logic Group Report Logic-87-9, Stanford University, 1987.

[Ginsberg and Smith, 1987b] Matthew Ginsberg and David Smith. Reasoning about action II: The qualification problem. Stanford Logic Group Report Logic-87-10, Stanford University, 1987.

[Kaelbling, 1988] Leslie Kaelbling. Goals as Parallel Program Specifications. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 60–65, St. Paul, Minnesota, August 1988. American Association for Artificial Intelligence.

[Minton, 1988] Steven Minton. Qualitative Results Concerning the Utility of Explanation-Based Learning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 564–569, St. Paul, Minnesota, August 1988. American Association for Artificial Intelligence.

[Nilsson, 1984] Nils Nilsson, editor. Shakey the Robot. Technical Note 323, Stanford Research Institute, 1984.

[Nilsson, 1988] Nils Nilsson. Action Networks. In *Proceedings of the Rochester Planning Workshop*, pages 21–52, Rochester, New York, October 1988. University of Rochester.

[Reisig, 1985] Wolfgang Reisig *Petri Nets: an Introduction.* Springer-Verlag, EATCS Monographs on Theoretical Computer Science, Volume 4, 1985.

[Sacerdoti, 1975] Earl Sacerdoti. The Non-Linear Nature of Plans. In *Proceedings of the Fourth Joint Conference on Artificial Intelligence*, pages 206–214, Tbilisi, Georgia, USSR, 1975. International Joint Committee on Artificial Intelligence.

[Schoppers, 1987] Marcel Schoppers. Universal Plans for Reactive Robots in Unpredictable Environments. In *Proceedings of the Tenth International Conference on Artificial Intelligence*, pages 1039-1046, Milan, Italy, 1987. International Joint Committee on Artificial Intelligence.

[Suchman, 1987] Lucy Suchman. *Plans and Situated Actions: the problem of human machine communication.* Cambridge University Press, 1987.

[Rosenschein and Kaelbling, 1987] Stan Rosenschein and Leslie Kaelbling. The Synthesis of Digital Machines with Provable Epistemic Properties. Technical Note 412, Stanford Research Institute, 1987.

[Tate, 1977] Austin Tate. Generating Project Networks. In *Proceedings of the Fifth Joint Conference on Artificial Intelligence*, pages 888-893, Boston, MA, 1977. International Joint Committee on Artificial Intelligence.

[Vere, 1981] Steven Vere. Planning in time: windows and durations for activities and goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume PAMI-5, No. 3, pages 246–267, 1981.

[Wilkins, 1984] David Wilkins. Domain independent planning: representation and plan generation. *Artificial Intelligence*, Number 22, pages 269–301, 1984.

Tractable Decision-Analytic Control

Oren Etzioni*
Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
ETZI@CS.CMU.EDU

Abstract

An autonomous agent's control problem is often formulated as the attempt to minimize the expected cost of accomplishing a goal. This paper presents a three-dimensional view of the control problem that is substantially more realistic. The agent's control policy is assessed along the three dimensions of deliberation cost, execution cost, and goal utility. The agent must choose which goal to attend to as well as which action to take.

This control problem is addressed in the context of an architecture with record-keeping and class-formation capabilities. Consequently, the architecture is able to offer its control module expected utility and expected cost estimates that are gradually refined as the agent accumulates experience. A programmer is not required to supply that knowledge, and the estimates are provided without recourse to distributional assumptions. Furthermore, control choices are made by a simple and fast algorithm. Thus, three of the standard objections to decision-analytic control are blocked.

The agent's control decisions are guided by our principle of greedy rationality—*choose the actions whose marginal expected utility (utility/cost) is maximal.* Thus, when necessary, the agent will prefer a speedy approximation to costly, high-utility alternatives.

1 Introduction and Motivation

In a paper dating back to 1975 Simon & Kadane defined the problem of satisficing search. Unlike best-value search—search aimed at finding the best goal node—satisficing search aims to minimize the *expected* search effort for reaching any of a set of goal nodes. Controlling the actions of a resource-bounded autonomous agent requires a three-dimensional view of search. To achieve satisfactory performance the agent must trade the utility of the sought goal (maximized by best-value search) against the expected cost of the path to the goal (minimized by satisficing search). The quality of that tradeoff must itself be traded against the agent's deliberation efforts, simply because the agent needs to act before it's too late. This paper proposes an approach to the control problem that leads to satisfactory behavior along the three dimensions of deliberation cost, execution cost, and goal utility.

1.1 A Critique of Simple Satisficing Search

Simon & Kadane introduce their approach with the fanciful example of searching for a treasure chest. Chests may be buried in any of a number of excavation sites. The agent's problem is to find a treasure chest as quickly as possible. Given expected cost and probability of success figures for the excavation of each site, Simon & Kadane derive a search strategy for the agent that provably minimizes the expected cost of the search. They proceed to generalize the strategy to the case where ordering constraints hold between different excavation operations. While mathematically pleasing, Simon & Kadane's formulation makes several important simplifications. It is the burden of this paper to remove some of these

*The conceptual framework presented above has been developed through numerous illuminating discussions with Tom Mitchell. Stuart Russell and Eric Wefald's lucid exposition of the issues involved in Decision-analytic control have inspired much of my thinking on the subject. Jonathan Amsterdam, Danny Sleator, Doug Tygar, and Raúl Valdés-Pérez have contributed to the formal aspects of this paper. Jim Blythe, Murray Campbell, Haym Hirsh, Craig Knoblock, Tom Mitchell, and Prasad Tadepalli provided helpful comments on a previous draft. Jonathan Amsterdam was of particular assistance in clarifying my notation. I am supported by an AT&T Bell Labs Ph.D. Scholarship. In addition, this research was sponsored by the National Science Foundation under contract IRI-8740522.

simplifications while retaining Simon and Kadane's original insight that minimizing expected search effort is an important component of the control problem.

Simon & Kadane presuppose that probability of success and expected cost figures are available for each potential search action. In practice, such figures must be computed. In section 3.1 we describe a mechanism that enables an agent to estimate these figures based on its problem-solving experiences. Although Simon & Kadane do not report on a complexity analysis of their procedure for computing an optimal search strategy, the procedure analyzes every node in the search graph. Consequently, the procedure's complexity is at least linear in the size of the graph. The search graphs in typical AI problems (e.g., chess) are sufficiently large that the computational cost of such a procedure is prohibitive. This paper proposes a tractable procedure for control.

Satisficing search does not distinguish between goals, nor does it distinguish between methods for achieving a given goal. But even in the simple treasure chest example, such additional complexity is warranted. The value of the treasures in different chests may vary, and distinct excavation procedures may yield varying portions of the treasures. Thus, a more powerful search control mechanism would not blindly seek to minimize search effort, but would rather trade execution effort and treasure value in an attempt to maximize the expected utility of the search subject to the agent's resource constraints. Computing which excavation procedure to use must not take too long, however, lest the agent run out of time. Thus, the three-dimensional view of search applies even to this simple case.

Satisficing search focuses on trying to achieve a single goal. An agent often has multiple independent goals, however. For example, there may be several islands with different treasures buried on each. The agent may wish to carry off as many treasures as it can before an impending native attack. Multiple independent goals are distinct from disjunctive subgoals, in that the agent wishes to achieve more than one of them. Multiple independent goals are also distinct from conjunctive subgoals in that failing to achieve one independent goal does not impinge on trying to achieve the rest. Our approach handles multiple independent goals.

1.2 Preview of the Paper

The control problem for a resource-bounded autonomous agent is essentially an economic problem—the problem of utilizing resources to maximize satisfaction. It is not surprising, therefore, that the central notions underlying our approach to the control problem, opportunity cost and marginal utility, are both borrowed from economic theory (Samuelson, 1976).

When the utilization of a resource (such as time or money) for some action A means that another action cannot be performed, A is said to have an opportunity cost. If several actions contend for the same resource, then the opportunity cost of choosing one of them is the maximum of the utility of the others. Attending graduate school, for example, has the opportunity cost of not being able to earn a higher salary in industry. The opportunity cost of time is at the crux of a time-bounded agent's control problem. Taking one course of action means that time may not be available to take another, and deliberating about which action to perform robs the agent of time that could have been used to act. Opportunity cost is defined precisely in Section 2.3.

The degree to which an agent desires a goal may be represented by defining a utility measure over world states. A state in which a goal is achieved has a higher utility than one in which it is not. In the treasure chest example, a simple utility measure is the value of the treasure the agent has on its ship as it departs. A utility function also models partial goal satisfaction. The utility of achieving a goal with a given method is a function of the goal's worth and the degree to which the method satisfies the goal.

Given a utility function, marginal utility may be defined as the derivative of the utility function with respect to a cost variable (time in this case). We employ marginal utility as a choice criterion between the options available to an agent. Picking the option whose marginal utility is greatest maximizes the ratio of expected utility to execution cost. Thus, an easy-to-obtain but small treasure chest may be preferred to a valuable but difficult-to-reach chest. Maximizing marginal utility enables the agent to trade goal utility for search time. It is the essence of *greedy rationality*, a notion that harks back to the conclusion of Simon & Kadane's forward-looking 1975 paper.

The balance of the paper is as follows: The next section formulates precisely the control problem introduced above. Sections 3 and 4 outline our solution to the problem. Section 5 analyzes two special cases of the general control problem that have proven amenable to precise theoretical treatment. Section 6 explores the benefits of using concept learning techniques in conjunction with decision-analytic control. And Section 7 discusses related work. The paper concludes by calling for the integration of decision-analytic and concept learning approaches to the control problem.

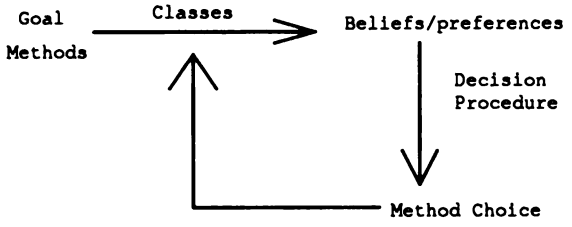


Figure 1: An Architecture

2 The Control Problem

In formulating and solving an agent's control problem, we will presuppose the architecture described below.

2.1 An Architecture

The architecture's default control strategy is depth-first search. Control knowledge obtained by the architecture is represented by (class, belief/preference) pairs. The class denotes the set of world states to which the belief or preference applies. Control rules (e.g., Laird *et al.*, 1987; Minton *et al.*, 1989) are roughly equivalent to this representation: a rule's left-hand side picks out the class to which the rule applies, and the rule's right-hand side expresses some belief or preference. Throughout the paper we refer to this set of classes as the agent's *state-division*. In chess, for example, dividing all possible board positions into opening, middle game, and endgame positions is a state-division. Although exhaustive, the classes in the state-division are not necessarily mutually exclusive.

In any given state, the agent must choose between the methods available to achieve its current goals. In our excavation example, methods may consist of digging and using dynamite. The class of "excavations into volcanic rock" may have the preference "use dynamite" associated with it. The architecture maps the current state to one or more classes in the state-division. Beliefs and preferences associated with these classes yield a (possibly empty) set of beliefs and preferences about executing the applicable methods in the current state. A decision procedure takes this set of beliefs and preferences as input, and outputs a method choice that is executed by the architecture and results in an updated state. A schematic depiction of the architecture appears in figure 1. The description models the current state of the Theo architecture (Mitchell *et al.*, 1989). Theo is a software framework intended to support the construction of autonomous agents.

Given:

- **A time constraint.**
- **Multiple goals.**
- **A set of methods for each goal.**
- **A Utility measure over final world states.**

Determine: **A sequence of methods that maximizes the expected utility of the agent's actions.**

Table 1: The General Control Problem

2.2 Formulating the Control Problem

Table 1 outlines an agent's control problem in decision-theoretic terms. An optimal solution is one that maximizes the agent's expected utility. Formulating the problem in these terms does not commit us to employing decision-analytic control, or to building an agent that explicitly considers the expected utilities of its options. *The problem formulation is external to the agent.* It may be used to model a device as simple as a thermostat or as complex as the Soar architecture, neither of which employ decision-analytic control.

2.3 Precise Formulation

Let B denote a deadline. Let the set of goals Γ be a subset of the set of final states F which is a subset of the set of world states S. In chess, for example, Γ denotes the set of board positions in which the agent has won the game, F denotes the set of final board positions in which the agent has either won, lost or drawn, and S denotes the set of legal board positions.

A method m is a (possibly atomic) sequence of actions that maps that current state s to a final state $m(s) \epsilon F$. $T_s(m)$ is the time which m takes to execute from state s. The utility measure U is defined over final world states. In chess, it corresponds to an evaluation function that is only defined over board positions in terminated games. Applying a method in a state s may result in failure, a final world state with zero utility. Furthermore, the utility of any final world state that is reached after the deadline B is 0. $P(m)$ is the probability that m terminates in a final world state with nonzero utility. Strictly speaking both U and P are functions of time, but our notation does not reflect this. From U we derive a utility function $\hat{U}$ for methods: $\hat{U}_s(m) = U(m(s))$.

Executing the method m from a variety of different world states will take differing amounts of time and result in one of several final world states. Thus, we define the expected utility of m as $E[\hat{U}(m)]$, and

the expected time consumption of m as $E[T(m)]$. The expectation is over the probability that the agent will encounter a given state.

To solve the control problem in Table 1 the agent chooses a sequence of methods σ. The first subscript of each method denotes the method number, and the second subscript denotes which of the agent's goals it is intended to achieve.

$$\sigma = (m_{1,1}, m_{2,1}..., m_{k_1,1}, m_{1,2}, m_{2,2}, ..., m_{k_n,n})$$

If the first method aimed at achieving goal g_1 fails, the agent moves to the second method and so on. Once goal g_1 is achieved the agent skips over any unexecuted methods aimed at achieving g_1, and tries to achieve goal g_2 and so on until the deadline B is reached.

The expected utility of σ is:

$$\begin{aligned} E[\hat{U}(\sigma)] = E[\hat{U}(m_{1,1})] + E[\hat{U}(m_{2,1})](1 - P(m_{1,1}))... \\ +E[\hat{U}(m_{1,2})] + E[\hat{U}(m_{2,2})](1 - P(m_{1,2}))... \\ +... \\ +E[\hat{U}(m_{1,n})] + ... + E[\hat{U}(m_{k_n,n})] \prod_{i=1}^{k-1}(1 - P(m_{i,n})) \end{aligned}$$

The above definition presupposes that the methods' expected utility and probability of success are independent of the order in which they are executed.

The framework introduced above enables us to define an optimal solution to the agent's control problem: an optimal solution is a method sequence whose expected utility is maximal. We denote such a sequence by σ^*_B because it is a function of the agent's deadline. We define two additional quantities required to state Proposition 1 below. First, the expected opportunity cost of a method m is:

$$E[\gamma_B(m)] = E[\hat{U}(\sigma^*_B)] - E[\hat{U}(\sigma^*_{B-T(m)})]$$

That is, the difference between the expected utility of the optimal method sequence before m is executed and the expected utility of the optimal method sequence after m. Second, the expected gain of a method m is its expected utility minus its expected opportunity cost:

$$E[G_B(m)] = E[\hat{U}(m)] - E[\gamma_B(m)]$$

We treat both gain and opportunity cost as random variables whose expectation is defined over the population of world states encountered by the agent.

2.4 Simplifications

The above formulation makes several simplifications. The control problem is defined for an off-line control mechanism. No analysis is provided of how to dynamically modify the flow of control at execution time. The only resource we consider is time, and the time constraint is a deadline. We have abstracted away from considering the cost of switching between different goals and the cost of recovering from failed methods. The model is easily elaborated to handle these considerations.

The formulation can also be extended to handle choices between the individual actions that make up methods. Each action has a set of *completions* M—the set of methods that begin with that action. An action may be "completed" differently in different states. A probability distribution over the completions of an action may be derived from a probability distribution over the states encountered by the agent. The expected utility of an action a is a function of the probability of its completions and their expected utility.

$$E[\tilde{U}_a] = \sum_{m \in M} p(m) E[\hat{U}_m]$$

The expected cost of an action a can be defined analogously. The necessary extensions to the record keeping mechanism are described in Section 3.1. Thus, the MU heuristic could be extended to choose between actions.

A more difficult problem is the assumption of mutual independence between the different methods and the different goals. This assumption is unrealistic in many domains. Multiple goals will often share subgoals and interact in various ways. The failure of one method may implies that other methods will fail etc. The performance of decision-analytic control mechanisms degrades in the presence of complex interactions.

2.5 Optimal Control

We can now define a criterion for choosing between methods that enables the agent to construct the optimal method sequence.

Proposition 1 *Repeatedly choosing the method whose expected gain is maximal enables the agent to construct an optimal method sequence.*[1]

The above proposition indicates how to directly map our problem formulation into a control cycle for an autonomous agent. This mapping fails because computing the expected gain of a method turns out to be intractable. Section 5.2 demonstrates that even in the case where only one method is associated with each goal, computing the optimal control strategy turns out to be NP-hard. The agent must therefore resort to a heuristic approach in order to address the control problem with manageable deliberation

[1] Proofs of the results in the paper appear in (Etzioni, 1989).

costs. For this reason, attempts to apply decision-theoretic ideas to the problem of control are forced to employ approximations to decision-theoretic formulae. We refer to such approaches as *decision-analytic* approaches to control.

The three-dimensional view of control is not explicitly represented in Table 1. However, the presence of multiple methods for satisfying (to varying degrees) multiple goals imply a range of possible execution cost and goal utility tradeoffs. The presence of a time constraint means that time spent deliberating is time that is not available to act. Thus, all three dimensions are present in our formulation of the control problem.

We do not assume that the agent is given expected utility and expected cost estimates for its methods. The following section describes how the agent acquires these estimates over time.

3 Acquiring Control Knowledge

3.1 Keeping Records

Initially, the agent proceeds by attending to its goals and executing its methods in an arbitrary default order. After the execution of a method, the agent records how long the execution took, and the utility obtained from the execution. If the method is an excavation procedure, for example, a clock is consulted before the procedure begins and after it terminates to determine its time cost. The utility of the excavation is determined by evaluating the utility of its result. The result may be a pile of gravel (with utility 0) an unmolested treasure chest (with utility equal to the value of the chest), or a portion of the buried treasure (with utility corresponding to the portion's value).

By definition, a method terminates in a final world state whose utility may be evaluated. A more elaborate sampling mechanism is required in order to extend our formulation to individual action choices. To record execution cost, a clock is consulted before each action is executed and, again, after the method (a sequence of actions) terminates. The time cost recorded for each action is the termination time minus the initial time, which is the cost of the portion of the method beginning with the given action. The utility of each action is simply the utility of the method which includes the action. In chess, for example, the utility of the final board position is assigned to each move made during the game.

A difficult credit assignment problem lurks here, because the game may have been won through a stroke of genius in the twentieth move even though the tenth move lost the queen. Yet the tenth move will be rewarded all the same. We do not have a satisfactory domain-independent solution to this problem. Nevertheless, adequate sample sizes will reduce its effect. It seems reasonable to suppose that the majority of games where a player loses his or her queen will be lost. Thus, although the treacherous tenth move has been assigned high utility in one instance, in other games its utility will be recorded more accurately. As section 3.3 shows, averaging over a sufficiently large sample of games is likely to lead to a reasonable estimate of the move's expected utility.

3.2 From Samples to Expectations

The agent's state-division defines classes of world states. The agent encounters states from each class with some probability (unknown to the agent). Hence, we may think of a given class in the state-division as a population. A sample of previously encountered instances is associated with each class.

Suppose, for example, that one of the agent's classes is "excavations in granite rock." The agent is concerned with estimating the cost of digging into the granite rock in a particular case. The mean or expected cost for the class can serve as a cost estimate for a particular instance. The quality of the estimate depends on the homogeneity of the class. If the class is perfectly homogeneous, for example, the mean will provide perfectly accurate estimates. If the variance of the class is high, on the other hand, the mean will be a poor estimator. A formal analysis of these considerations is provided in (Etzioni, 1988).

3.3 Distribution-Free Estimation

Of course, the mean of the class is not known to the agent. However, the record-keeping apparatus described above associates cost and utility information with instances of method executions. In turn, these instances are associated with the classes in the agent's state-division yielding samples from which the population means may be estimated. The distance of a sample mean, $\bar{X}$, from the population mean, μ, is a rapidly decreasing function of the sample size. Hoeffding's inequality (Hoeffding, 1963) quantifies this intuition.[2]

$$p(\mid \bar{X} - \mu \mid \leq \epsilon) \geq 1 - 2e^{-2n\epsilon^2/(b-a)^2}$$

The inequality provides us with a lower bound on the probability that $\bar{X}$ is within ϵ of the true mean *without making distributional assumptions*. The range of the random variable is assumed to be bounded, however. Hoeffding's inequality also presupposes a

[2] X is an independent random variable, $a < X < b$; $\bar{X}$ is the mean of a random sample of X; n is the sample size.

fixed (but arbitrary) probability distribution over the population.

Hoeffding's inequality enables us to compute the sample size required to guarantee bounds on the reliability and accuracy of estimated means. As the following definition makes apparent, the required sample size increases logarithmically with the desired reliability and quadratically with the desired accuracy for a mean estimate.[3]

Definition 1 $sample\text{-}size(\delta, \epsilon) = \frac{(b-a)^2}{2\epsilon^2} ln(\frac{2}{\delta})$.

If the value of *sample-size* is substituted for n in Hoeffding's inequality, the resultant inequality is $P(\mid \bar{X} - \mu \mid \leq \epsilon) \geq 1 - \delta$. Therefore,

Proposition 2 *Given values for any pair out of the triple ϵ, δ, and sample-size, the value of the third parameter may be computed via Hoeffding's inequality.*

The analysis in Section 5 makes use of this proposition to determine the accuracy of its estimates given samples and desired reliability levels. Expected accuracy and reliability are derived for populations of states. The statistical theory used provides no guarantees on the error of using the estimates to make predictions in any single state.

3.4 Class Formation

Hoeffding's inequality demonstrates that as the agent accumulates experience (which is translated to increased sample sizes) the reliability and accuracy of its estimates increase. At any given point, however, the agent may not be certain of having a large sample at its disposal for reliably estimating the mean of a given class. The agent may, therefore, combine several related classes (e.g., digging in granite and digging in basalt), compute the sample mean for the combined classes, and use that sample mean as its estimate for new instances from the combined classes. The likelihood that the agent will have an adequate sample size increases with the size of the class. However, as more classes are combined into one it is also likely that the variance of the class will grow thus reducing the accuracy of the estimate. Consequently, the agent must trade sample size and class variance against each other.

Although the agent is likely to succeed in obtaining reliable and accurate mean estimates for sufficiently large populations, the agent's choice of population or class is critical for the *usefulness* of the estimates. Suppose that the agent is trying to choose between digging into the granite rock and using dynamite to clear the ground. If the agent were to map the current excavation instance to the class of excavations performed on weekends, for example, dynamite would be unlikely to distinguish itself over digging. In some cases digging is superior to dynamite and there is no correlation between the cases and the class of "weekend excavations." In contrast, the estimated figures will be useful if the agent succeeds in mapping the instance to a homogeneous class such as "excavations in volcanic rock."

Homogeneous classes are often required to provide accurate cost and utility estimates. Randomly aggregating instances into classes is unlikely to prove beneficial. It is unrealistic to expect, however, that a domain-independent control mechanism will broach a new domain with the appropriate classes already formed. *Therefore, dynamic class formation is essential to any domain-independent instantiation of the model described in Figure 1.* The agent can employ Concept learning techniques to dynamically form homogeneous classes based on its past experience.

[3] Results obtained by (Haussler, 1988) in a concept learning application suggest that this bound can be improved upon.

1. For each method (of each goal):
   (a) Map the (state, method) pair to a class of states.
   (b) Collect previously encountered instances of this class into a sample.
   (c) Compute the mean utility $\bar{U}$ and mean cost $\bar{T}$ of the sample.
   (d) Form the method's key: $\bar{U}/\bar{T}$.
2. Sort the methods on their keys in decreasing order.
3. Execute the methods in succession.

Table 2: A (simplified) Control Cycle

4 A Greedy Control Cycle

The previous section described how an agent can, over time, acquire and refine utility and cost estimates for its methods. Our proposed control mechanism utilizes these estimates to guide the agent. A simplified description of the control cycle appears in Table 2. In a nutshell, the proposal is to choose methods in a best-first fashion where the evaluation function is the methods' marginal expected utility. In accord with the abstract description in Figure 1, the cycle maps the current state to classes of states and uses beliefs about these classes as inputs to a decision procedure. The decision procedure chooses the next method. Specifically, the agent maps each (state , method) pair to a class of states. A sample

of previous executions of the method (in states belonging to the class) is indexed under the class. The agent attempts to find a homogeneous class which has a sufficiently large sample associated with it. The mean utility and cost of the sample are computed and used as estimates or predictions of the cost and utility of executing the given method in the current state. If the sample associated with the class is too small to yield reliable predictions, a search may be initiated for a larger class with an adequate sample.

Once all the cost and utility estimates have been computed, the methods are sorted in decreasing order on the ratio of their individual utility/cost estimates. That is the essence of our decision procedure. As we discuss in Section 6 the possibility of refining the state-division is considered. If it is rejected the sorted methods are executed in succession, otherwise the state-division is refined and the control loop is restarted.

We have glossed over numerous details to convey the essence of the control cycle in limited space. Execution considerations such as updating time bounds, removing redundant methods from the method list, and the handling of subgoals, as well as efficiency considerations such as caching sample means are not discussed.

4.1 The Decision Procedure

For the sake of tractability, our decision procedure sorts the available methods on estimates of their marginal expected utility rather than on their expected gain. Marginal utility is the derivative of the utility function with respect to the cost variable (time in our case). Utility in our model is not a continuous or differentiable function since executing a method has value only once its goal is achieved. However, dividing the utility of executing a method by the time it takes is an easily computed, discrete analog to the continuous case. Of course, this computation is done for all the methods before their execution in order to choose the method to execute. We treat the utility and cost of a method about to be executed as random variables. The values used, therefore, are estimates of the expected cost and utility of the outcome of each method's execution. These estimates are sample means derived from past experience as outlined in Section 3.1.

Maximizing the marginal expected utility is a greedy control heuristic. The agent picks the method that is expected to maximize the "return" for its time investment. We refer to this approach as greedy rationality, and to the heuristic used as the MU heuristic. Employing the MU heuristic enables the agent to trade goal utility (and degree of satisfaction) for reduced execution cost. Thus, when necessary, the agent will prefer speedy approximations to costly, high-utility alternatives. For example, the agent will prefer a treasure worth one thousand dollars, which can be found in five minutes to a treasure worth three thousand dollars, but which requires one hour to excavate. However, the agent will prefer a sixty thousand dollar treasure, which requires an hour to excavate, to the five-minute treasure above.

5 Formal Analysis

What guarantees can we make on the performance of an agent using the MU heuristic? In two special cases described below we have demonstrated that the heuristic is optimal (in the single goal case, Section 5.1) or within a factor of two of optimal (in the single method case, Section 5.2). The suboptimality of decisions made by the heuristic is shown to decrease linearly with the inaccuracy of the estimates used. In the general case, Section 5.3 guarantees that the heuristic exploits dominance relations between methods. Thus, if one option appears better than another in every regard the MU heuristic will provably make the correct choice. The problem of further analyzing the MU heuristic is open.

5.1 The Single Goal Case

The MU heuristic may result in suboptimal performance if the agent attempts to achieve a single goal given multiple methods and a deadline.[4] However, a slightly altered model demonstrates the efficacy of the MU heuristic. Consider the case where the agent attends to a single goal, but has a constant opportunity cost γ for every time unit spent on the goal. Since the agent doesn't know which of its methods will succeed in the current state, it attempts to maximize the expected gain of its actions. We can model the agent's control problem as follows:

Given:

- A current goal g.
- An opportunity cost for time γ.
- A set of methods M for achieving the goal g.
- Each method $m \epsilon M$ has
  - An expected time cost $E[T(m)]$.
  - An expected utility $E[\hat{U}(m)]$.
  - A probability of achieving a satisfactory outcome $P(m)$.

Determine: A method ordering whose expected gain is maximal.

[4] Consider the case where the expected utility for a method m is the same as m's expected cost, and this holds for all methods.

The expected gain of a method m is:

$$E[G(m)] = E[\hat{U}(m)] - \gamma E[T(m)]$$

The agent only executes methods whose expected gain is positive. Let σ stand for an ordering of the method set M indexed by i, and define $P(0) = 0$. The quantity to be maximized is:

$$E[\hat{U}(\sigma)] = \sum_{i \text{ s.t. } E[G(m_i)] \geq 0} E[G(m_i)] \prod_{k=0}^{i-1} (1 - P(m_k))$$

Thus, a method ordering is optimal if it maximizes the expected gain of trying to achieve the current goal. This definition assumes that the methods' probability of success, utility, and cost are independent of the order in which the methods are executed.

Naively, the optimal method ordering can be found by computing the expected gain for all possible method orderings, but for N methods this takes O(N!) time. Proposition 3 suggests an equivalent O(NlogN) procedure.[5]

Proposition 3 *Sorting M on $E[G(m)]/P(m)$ in decreasing order results in an optimal method ordering.*

The cases in which time has no cost for the agent can be modeled by setting γ to 0. In that case, maximizing the expected gain reduces to simply sorting the methods on their maximal utility outcomes. Neither time consumption nor probability of success have any relevance in this case, because the agent has unlimited time at its disposal, at no cost.

Of course, this is highly unrealistic. As our own experience indicates, an agent always has a positive opportunity cost for time. There is always something useful to do. Previous formulations of the control problem have focused on minimizing the expected time for achieving the goal given a set of equal-utility methods with binary outcomes (either success or failure) and varying costs. This formulation is easily accommodated within our framework by making some additional simplifying assumptions. Namely, setting γ to 1 and assuming that $U(m)$ is identical for all methods m in M.

Under these assumptions the key reduces to:

$$[P(m)U(m) - E[T(m)]/P(m)$$

The $U(m)$'s may be factored out since they are the same for all methods leaving us with $-E[T(m)]/P(m)$ or equivalently $P(m)/E[T(m)]$, which is a result obtained by (Simon and Kadane, 1975; Barnett, 1984; Smith, 1988) and others. Under the stated assumptions, sorting methods on this key is equivalent to sorting the methods on their marginal expected utility. Thus, *employing the MU heuristic leads to optimal behavior in this special case.*

In practice, we do not know $E[T(m)]$ and $P(m)$ for each method, but these parameters can be estimated from samples as discussed in Section 3.2. Let $\bar{P}(m)$ and $\bar{E}[T(m)]$ be estimates of $P(m)$ and $E[T(m)]$ such that

$$p(|\bar{P}(m) - P(m)| \leq \epsilon) \geq 1 - \delta$$

and

$$p(|\bar{E}[T(m)] - E[T(m)]| \leq \alpha) \geq 1 - \delta$$

That is, mean estimates whose accuracy and reliability is guaranteed by Hoeffding's inequality. Let the number of methods for achieving a given goal be N. Define σ^h to be the method ordering produced by sorting the methods on $\bar{P}(m)/\bar{E}[T(m)]$, and let σ^* be the optimal method ordering. Define the time cost of a sequence, $T(\sigma)$, to be the sum of the time cost of the elements of σ.

Proposition 4 *$E[T(\sigma^h)] - E[T(\sigma^*)]$ is linear in $1/\epsilon$ and $1/\alpha$ with probability linear in $1/\delta$.*

That is, the expected cost of the method ordering produced by sorting on the estimates provided diverges from the optimal expected cost by an amount that decreases linearly with the inaccuracy of the estimates, with probability that grows linearly with the reliability of the estimates. A similar result was obtained by Barnett (1984).

Our more general framework enables us to make the following observations:

- The expected gain of executing the methods can be recomputed after one or more methods have failed. When the expected gain drops below 0, the agent ought to abandon the current goal.
- When $\gamma > 1$, the agent ought to trade the degree of satisfaction on the current goal for time to be used on other goals. $\gamma > 1$ indicates that there is "pressing business" at hand.
- Similarly, when $\gamma < 1$, the agent ought to proportionally discount the time cost of satisfying the current goal, because the opportunity cost for time is low.

The following section argues that opportunity cost for time γ cannot be tractably computed in general. Indeed, previous work has implicitly assumed it to be always equal to one. The above observations are of interest for special cases in which the opportunity cost is readily available.

5.2 The Single Method Case

Given a set of multiple independent goals, a single method for achieving each goal whose success

[5] The number of possible outcomes for each method is assumed to be bounded by a constant.

is guaranteed, and a deadline—the agent's (specialized) control problem is isomorphic to the knapsack problem. We can define the optimization version of the knapsack problem as follows:
Instance: A finite set I, for each $i \epsilon I$ there is a weight $w(i) \epsilon Z^+$, a value $v(i) \epsilon Z^+$, and a positive integer "weight capacity" B. $\forall i \epsilon I, w(i) \leq B$.
Question: What is the subset $I' \subseteq I$ such that $\sum_{i \epsilon I'} v(i)$ is as large as possible, subject to the constraint $\sum_{i \epsilon I'} w(i) \leq B$?

In the control case the items are methods. The value of an item may be interpreted as the utility of achieving a goal with the given method; the weight of an item becomes the time required to execute the method, and the weight capacity of the knapsack is a deadline. The knapsack problem is known to be NP-hard. Since computing the expected opportunity cost of a method necessitates computing the optimal solution to a knapsack problem, *computing the expected opportunity cost of a method is NP-hard.* Furthermore, while estimates of expected utility and expected time consumption are readily produced by the mechanisms in Section 3, estimates of opportunity cost are not. *Thus, an agent facing multiple goals and a deadline will not be able to tractably determine its optimal course of action.*

However, powerful approximation algorithms exist for the knapsack problem. Garey & Johnson (p. 135) remark that it is not difficult to show that the following approximation algorithm comes within a factor of two of optimal:

1. Pick the item in I whose value is maximal and call it *max*.
2. Sort the items in $i \epsilon I$ in decreasing order on their value-density $v(i)/w(i)$.
3. Starting with I' empty, proceed sequentially through I, each time adding the next $i \epsilon I$ to I' whenever the sum of the weights of the items already in I' does not exceed $B - w(i)$.
4. Compare the combined value of the elements in I' with $v(max)$ and take the better of the two.

Essentially, the algorithm sorts the items on their value-density and places them in the knapsack in that order. This scheme is isomorphic to step 2 in our control cycle (Table 2).[6] Marginal utility is equivalent to value-density. We use the term marginal utility to underscore the connection between our approach and economic theory.

Two steps must be taken before Garey & Johnson's guarantee on the performance of their algorithm can be mapped to our single method case. First, since the actual value and weight figures are not known for our items (the methods), Garey & Johnson's guarantee must be shown to hold for expected value and weight figures. Let σ^h be the packing produced by the value-density algorithm run on expected value and weight figures, and Let σ^* be the packing of the knapsack whose expected value is maximal. Define the value of a sequence, $v(\sigma)$, to be the sum of the values of the elements of σ.

Proposition 5 $2E[v(\sigma^h)] - E[v(\sigma^*)] \geq 0$

The value-density algorithm's packing is within a factor of two of optimal. As in the single goal case, this proposition does not suffice because we only have estimates for the expected utility and time cost for the methods. The statistical theory presented in Section 3.3 demonstrates that we can control the divergence of our estimates from the true expected value by using sufficiently large samples. However, is the suboptimality of the value-density algorithm a well-behaved function of estimates' accuracy? The answer is yes.

Let $\bar{v}$ and $\bar{w}$ stand for the expected weight and expected value estimates whose inaccuracy is bounded by ϵ with probability at least $1 - \delta$ as guaranteed by Hoeffding's inequality. Let N be the number of items in the set I. Suppose that for any set of items $I' \subseteq I$ we have $\bar{w}(I') \leq B$ if and only if $w(I') \leq B$. That is, no item is excluded from a packing due to the inaccuracy in the estimation of its weight.[7] The suboptimality of the packing produced by the value-density algorithm is linear in $1/\epsilon$ with probability that is linear in $1/\delta$.

Proposition 6 $2E[v(\sigma^h)] - E[v(\sigma^*)] + 2N\epsilon \geq 0$ *with probability at least* $1 - 2N\delta$.

Again, the MU heuristic proves to have satisfactory and well-understood performance.

5.3 Exploiting Dominance

The previous subsections covered two important special cases: the single method (multiple goals) case, and the single active goal (multiple methods) case. This section offers a weak but important guarantee on the behavior of the MU heuristic in general.

A method is said to *dominate* another method for achieving a given goal if, according to the agent's beliefs, the dominating method is no worse along any dimension, and better along at least one. For example, if method A is cheaper, has higher utility, and is more likely to succeed than method B, then method A dominates method B. If we consider $>, <, =$ as three possible qualitative relationships between two

[6] Our algorithm also compares the maximal value item with the sorted list, but this detail is glossed over in the in Table 2.

[7] In practice we can guarantee that this assumption holds by introducing a small leniency in the deadline.

methods along a given dimension, then we may observe fourteen distinct (up to symmetry) qualitative relationships between methods compared on their cost, probability of success, and utility.

Of the fourteen, seven represent dominance relations (in one remaining case the methods are interchangeable and the other six represent tradeoffs). This observation gives credence to Wellman's assertion that "the ability to separate tradeoffs from obvious choices defines a lower bound on competence for decision-making agents" (Wellman, 1988, page xvi). It is indeed important to guarantee that a control heuristic exploits dominance relations.

Proposition 7 *If one method dominates another, the MU heuristic will choose the dominating method.*

6 Optimal Versus Maximal

Section 5 demonstrated that a record-keeping agent can rapidly converge to optimal (or close to optimal) control choices in two special cases of the general control problem. Surprisingly, perhaps, optimality is not a very strong guarantee on the performance of an agent. Optimal control merely means making the choice that is expected to be the best over a class of cases. Thus, if the agent's state-division is too coarse poor performance will ensue.

Consider an agent that has no state-division and two actions at its disposal. Based on its experience the agent determines that the probability of achieving its goal is 0.6 using action A, and 0.5 using action B. Clearly choosing action A is optimal, because on average choosing action A is better than action B. However, always choosing action A will still result in poor performance. If the agent may only use one action, optimal control will lead to success merely sixty percent of the time. It is possible, however, for the agent (given a better state-division) to succeed one hundred percent of the time.

In order to measure how 'good' optimal control is, we may define a notion of *maximal* control. A maximal control mechanism chooses the option that leads to the maximal utility outcome in any given world state. It is unlikely that an agent will be able to always find the maximal choice. However, defining the notion of maximal control provides us with a vantage point from which to evaluate the performance of an optimal control mechanism.

6.1 Improving on Optimality

An agent may improve on the optimal control strategy by refining its state-division. The application of concept learning techniques to acquiring control knowledge (e.g., Minton *et al.*, 1989; Mitchell *et al.*, 1989) may be analyzed in these terms. Unfortunately, the problem of quickly acquiring general and effective control knowledge is difficult (cf., Etzioni and Mitchell, 1989). In this section we consider how our architecture may be used to accelerate the convergence of an agent towards maximal control.

Standard concept-learning approaches to control attempt to define classes of states where applying a given method is guaranteed to lead to success. This amounts to trying to define equivalence classes over world states. Certainly, the best possible state-division is a set of equivalence classes over states with respect to the quantities of interest—utility, cost, and probability of success. Even in very simple domains, however, such state-divisions are difficult to acquire. Fortunately, equivalence classes are sufficient *but not necessary* even for maximal control. In fact, any state-division that results in the same method ordering as the equivalence-class division is maximal as well.

For example, if method A's time cost ranges from 100 to 200 seconds, and method B's cost ranges from 500 to 800 seconds, then method A will be preferred to method B, *ceteris paribus*. Obtaining precise predictions of the cost of A and B is unnecessary since method A will always be cheaper. Although maximal divisions are difficult to acquire, the above example demonstrates that approximate information can be used to make maximal control decisions.

6.2 Controlling Refinement

A time-bounded agent is faced with the problem of allocating some of its scarce time to refining its state-division. Time spent refining the state-division is time that is not available to act. The agent must, therefore, trade the time saved by having a superior state-division against the time spent refining the division. This tradeoff may be formulated precisely in decision-theoretic terms (cf., Barnett, 1984). Consider the two methods F and G whose expected time costs are 500 and 600 seconds respectively. *Ceteris paribus,* method F will be chosen. The expected loss of always choosing F, relative to making the maximal choice at each point, can be estimated from a sample. The estimated loss of using the current state-division enables an agent to decide whether to allocate resources to refining the state-division or not. Comparing the estimated loss of making control choices based on different classes in the state-division tells the agent which classes to refine. Thus, the process of judiciously refining the state-division over time is facilitated. Unfortunately, the problem of mapping estimated time savings from refinement to their utility for the agent remains unsolved.

7 Related Work

In the AI literature Simon & Kadane (1975) provide an early example of the use of decision-theoretic ideas to control search. More recently, Smith (1988) has extended Simon & Kadane's approach to the problem of controlling backward inference. Although important, this body of work suffers from three chief drawbacks: The control problem is solved for a single goal, the decision-making computations are at least linear in the size of the given search graph, and no indication is given as to how to obtain or estimate the requisite knowledge. The approach developed by Simon & Kadane and Smith is more general than our own because it provides guidance for individual action choices. However, we have indicated how to extend our approach in this direction.

Sproull's dissertation (1977) introduced a three-dimensional view of planning. Sproull argued that finding an optimal plan is often suboptimal when the costs of planning are taken into account. He pointed to the possibility of employing decision theory to trade execution cost, plan reliability, and goal utility. Sproull continued to assume that the requisite knowledge for the planner's decision analysis is given.

The work of (Natarajan, 1988) on optimizing Prolog programs, and of (Rendell, 1983; Abramson and Korf, 1987; Lee and Mahajan, 1988) on constructing evaluation functions for game-playing domains represented the next step forward by suggesting that sampling a program's experience can allow it to estimate the extensive knowledge required for decision-analysis.

Abramson & Korf describe an application of decision-theoretic ideas to constructing evaluation functions for game-playing. Their approach estimates the expected utility of making a move by randomly sampling continued play after that move. The estimated utility is used as the move's evaluation. Despite being developed independently, Abramson & Korf's approach is similar in spirit to our own. The application to game-playing enables Abramson & Korf to ignore execution cost. Hence, they sort their options on estimates of expected utility rather than *marginal* expected utility as in our approach. Abramson & Korf's work demonstrates the deep connection between search guided by heuristic evaluation functions and decision-analytic control.

Russell & Wefald (1988) present a general theory of decision-analytic control. Russell & Wefald propose to control an agent's computations by estimating their expected utility. They point out that the utility of an agent's computations depends on their effect on the agent's ultimate action choices. The link between the computations performed and the agent's choice of actions depends on the specific architecture being studied. Russell & Wefald have applied their general theory to several problems including the control of game-tree expansion with promising empirical results. Like ours, their approach relies on sampling to obtain estimated values that are fed into a simple and fast decision procedure. Furthermore, both approaches converge to optimal control with increased sample sizes, and both require a state-division into homogeneous classes to achieve satisfactory performance.

In contrast to our approach, however, their application to game-tree expansion employs powerful meta-level theorems to move from estimates for the values backed up from leaf expansions to utilities for the agent. To simplify their analysis Russell & Wefald explicitly make several assumptions. The most interesting from our perspective is the assumption of a uniform opportunity cost for time. This assumption allows Russell & Wefald to focus on computing the expected *utility* of computations instead of the expected *gain*, which is difficult to compute (as argued in Section 5.2).

Analytic models of previous work have been provided at the cost of making distributional assumptions. Russell & Wefald and Lee & Mahajan assume a normal distribution, and Smith assumes a uniform distribution. Our approach is unique in employing distribution-free statistical theory to analyze our control mechanism. In fairness, when distributional assumptions are empirically validated (e.g., Lee and Mahajan, 1988) they can allow more efficient sampling. However, since the statistical inference required is merely the estimation of means, an inference for which adequate distribution-free theory exists, it seems more parsimonious to dispense with distributional assumptions in the general case.

8 Concluding Remarks

The opportunity cost of time has emerged as a paramount consideration in choosing a course of action that is satisfactory along the three dimensions of deliberation cost, execution cost, and goal utility. As demonstrated by our analysis of the knapsack problem, computing the opportunity cost of time is intractable. Thus, an agent facing multiple goals and a time bound will not be able to tractably determine its optimal course of action. Our solution is to employ the principle of greedy rationality (choose the action whose marginal expected utility is maximal) as the basis of our control regime. Happily, greedy rationality obviates the need for computing or even estimating the opportunity cost of time.

Previous work on decision-analytic control has not considered the problem of acquiring a division of an

agent's world into classes. Section 6 argues that acquiring and refining such a division is essential for satisfactory performance. Simply, finding the "optimal chess move" will not do. The 10^{40} or so chess positions have to be divided into classes for which optimal moves can be determined. Several architectures have applied concept learning techniques to the control problem with some success. Considering the chess domain, however, illustrates the difficulty of the class-formation task for realistic applications.

Solving the class-formation problem is facilitated by the use of a decision-analytic control mechanism with record-keeping capabilities. The decision-analytic control mechanism exploits approximate information in the form of probabilities and estimated costs/utilities. Thus, even when the agent does not know what the optimal action is, it can quickly make an educated guess. In contrast, attempts to actually compute the identity of the optimal action suffer from a prohibitive deliberation cost. The record-keeping module enables an agent to evaluate the classes it has formed. If a class turns out to be nonhomogeneous, it may be refined. Thus, feedback from the record-keeping module can guide the class-formation process. Finally, learning which of several goals to attend to is a problem that has received scant attention in the machine learning literature. Our approach suggests how to reduce this problem to a standard class-formation or concept-learning problem.

References

Abramson, B. and Korf, R., 1987. A model of two-player evaluation functions. In *Proceedings of the 1987 National Conference on Artificial Intelligence*, Morgan-Kaufmann.

Barnett, J. A., 1984. How much is control knowledge worth? a primitive example. *Artificial Intelligence, 22*, 77–89.

Etzioni, O., 1988. Hypothesis filtering: a practical approach to reliable learning. In *Proceedings of the Fifth International Conference on Machine Learning.*

Etzioni, O., 1989. *Tractable Decision-Analytic Control: An Expanded Version.* Technical Report CMU-CS-89-119, Carnegie Mellon University.

Etzioni, O. and Mitchell, T. M., 1989. A comparative analysis of chunking and decision-analytic control. In *Proceedings of the AAAI Spring Symposium on AI and Limited Rationality.*

Haussler, D., 1988. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence, 36(2)*, 177–222.

Hoeffding, W., 1963. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association, 58(301).*

Laird, J. E., Newell, A., and Rosenbloom, P. S., 1987. Soar: an architecture for general intelligence. *Artificial Intelligence, 33*, 1–64.

Lee, K. F. and Mahajan, S., 1988. A pattern classification approach to evaluation function learning. *Artificial Intelligence, 36(1).*

Minton, S., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O., and Gil, Y., 1989. *Explanation-Based Learning: A Problem-Solving Perspective.* Technical Report CMU-CS-89-103., Carnegie Mellon University. To appear in *Artificial Intelligence.*

Mitchell, T. M., Allen, J., Chalasani, P., Cheng, J., Etzioni, O., Ringuette, M., and Schlimmer, J. C., 1989. Theo: a framework for self-improving systems. To appear in *Architectures for Intelligence*, K. VanLehn(ed.), Erlbaum, Hillsdale, NJ.

Natarajan, K. S., 1988. *Adaptive Search: An Approach to Optimize Search Effort for Problem Solving.* Technical Report 12228, IBM Watson Research Center.

Rendell, L., 1983. A new basis for state-space learning systems and a successful implementation. *Artificial Intelligence, 20.*

Russell, S. and Wefald, E., 1988. *Decision-Theoretic Control of Reasoning: General Theory and an Application to game-playing.* Technical Report UCB/CSD 88/435, University of California at Berkeley.

Samuelson, P. A., 1976. *Economics.* McGraw-Hill, 10th edition.

Simon, H. A. and Kadane, J. B., 1975. Optimal problem-solving search: all-or-none solutions. *Artificial Intelligence, 6.*

Smith, D. E., 1988. *Controlling Backward Inference.* Technical Report LOGIC-86-68, Stanford Logic Group. To Appear in *Artifical Intelligence.*

Sproull, R. F., 1977. *Strategy Construction Using a Synthesis of Heuristic and Decision-Theoretic Methods.* Technical Report CSL-77-2, Xerox PARC.

Wellman, M., 1988. *Formulation of Tradeoffs in Planning Under Uncertainty.* PhD thesis, M.I.T. Available as technical report MIT/LCS/TR-427.

A General Framework for Sorted Deduction: Fundamental Results on Hybrid Reasoning *

Alan M. Frisch †
Department of Computer Science
University of Illinois
1304 West Springfield Avenue
Urbana, IL 61801

Abstract

Researchers in artificial intelligence have recently been taking great interest in hybrid representations, among them sorted logics—logics that link a traditional logical representation to a taxonomic (or sort) representation such as those prevalent in semantic networks. This paper introduces a general framework—the substitutional framework—for integrating logical deduction and sortal deduction to form a deductive system for sorted logic. This paper also presents some results that provide the theoretical underpinnings of the framework. A distinguishing characteristic of a deductive system that is structured according to the substitutional framework is that the sort subsystem is invoked only when the logic subsystem performs unification, and thus sort information is used only in determining what substitutions to make for variables. Unlike every other known approach to sorted deduction, the substitutional framework provides for a *systematic* transformation of unsorted deductive systems into sorted ones.

1 Introduction

Recently, researchers in artificial intelligence (AI) have been taking great interest in hybrid representations—representation systems that consist of two or more integrated subsystems, each of which may employ distinct representation languages and inference systems. Included among such systems are:

- sorted logics (e.g., [McSkimin and Minker, 1979], [Cohn, 1987], and [Walther, 1987]), which integrate logical languages and languages for sort information,
- systems that mix assertional information and terminological information (e.g., KRYPTON [Brachman *et al.*, 1983] and KL-TWO [Vilain, 1985])
- systems that mix a weak logic with a taxonomic representation (e.g., [Frisch, 1986a] and [Patel-Schneider, 1987]),
- logic programming systems that integrate Horn-clause reasoning and inheritance (e.g., LOGIN [Aït-Kaci and Nasr, 1985] and HORNE [Frisch *et al.*, 1983]) and
- logics with built-in theories (e.g., theory resolution [Stickel, 1985]).

Researchers often cite two advantages to using hybrid representations. They point out that, instead of expressing all information in a single general-purpose representation language, it is often easier to express different kinds of information in representation languages specialised to each. Secondly, they claim that the use of specialized representations enables them to use specialized, and hence more efficient, methods of reasoning.

One particular form of hybrid representation that has been receiving a good deal of attention is sorted logic. Sorted logics can be seen as hybrid representations that link a traditional logical representation to a taxonomic representation such as those prevalent in semantic networks. The taxonomic component, which I shall call a sort module, contains information about relationships among various categories, or *sorts*. The logic component uses as its representation language a standard first-order predicate calculus that is augmented with sorted variables—variables that are restricted to taking on values that are in a specified sort.

In axiomatising a domain, many researchers—both in mathematics (e.g., Feferman [1974]) and in artificial intelligence (e.g., Hayes [1971; 1985], McDermott [1982] and Allen [1984])—have long preferred sorted logics to unsorted ones. The sorted logics are more natural because one usually wants to make a general claim about every individual in a certain class rather

*I thank the many people with whom I have had fruitful discussions related to this work, especially Tony Cohn. This work has been partially supported by NASA under grant number NAG 1-613 and by the University of Illinois Research Board.

†Phone: (217)-244-6024, Internet: frisch@cs.uiuc.edu, Usenet: uidcs!frisch

than every individual in the entire universe. However, only recently have researchers begun to explore the possibility of using the sorted logics in automated deductive systems. Some extremely powerful deductive systems for sorted logic have been built and, in several test cases, have demonstrated dramatic superiority over unsorted systems. Among these demonstrations are Cohn's [1985] and Walther's [1985] solutions to Schubert's Steamroller problem[1]—a puzzle-like deductive problem—and Frisch's [1986b] work on a deductive parsing problem.

Constructing a hybrid representation system involves more than just building a multitude of components, each with its specialized representation language and specialized deductive methods. The various deductive systems must be integrated in a manner that enables the knowledge represented and deduced by one component to be available for other components to use. Even if each component is a complete deductive system, the entire system may be incomplete if the components are not integrated in a proper manner.

A common way for two deductive components to interact is for one component to use some special form of unification that invokes the second deductive system as a subroutine. In some, though not in all, systems this is the only use made of the information stored in the second component.

There are a number of sorted logics whose deductive systems are prime examples of this architecture. There are two ways in which sorts commonly enter a sorted logic: variables can be restricted to range over the elements of a given sort and predicates and functions can be restricted to take arguments of some given sorts. When performing unification in a sorted logic these restrictions can be used to eliminate certain substitutions from consideration. To do this the unifier calls upon a second deductive mechanism, the sort module, to decide whether certain sentences are consequences of the information that it has stored. For example, the sort module may store the information that Clyde is in the sort of elephants and that everything in the sort of elephants is in the sort of mammals. During unification the sort module may be asked to deduce that Clyde is in the sort of mammals.

Suppose that we are given an unsorted logical language with ordinary variables and a deductive system that operates on that language by using unification to handle variables. Now suppose that we extend the unsorted language to a sorted language and wish to extend the unsorted deductive system to operate on the sorted language. Furthermore, suppose we wish to do this repeatedly for many languages and deductive systems. Must we initiate a large research project to work on each deductive system individually or is there a set of principles and a systematic way to generate these new systems?

In all previous work, each system for sorted deduction and its completeness proof is produced from scratch. No general principles or methods have been available to do this systematically. For example, the recent theses of Cohn [1983; 1987] and Walther [1987] present sorted logics and develop resolution-based deductive systems for them. Neither of these deductive systems are generated systematically from deductive systems for unsorted logics. Consequently, each of these theses is forced to conclude by raising an open question: how can other unsorted deductive systems be extended to deal with a sorted logic?

This paper introduces a framework for integrating logical deduction and sortal deduction to form deductive systems for logics with sorted variables. In a deductive system constructed according to this framework the sort module is invoked *only* when the logic module performs unification. Because the sort information gets used only when making substitutions, this framework is called the *substitutional framework.*

The substitutional framework provides not only an architecture for sorted deduction, but also a method for systematically transforming unsorted deduction systems into sorted ones. As we shall see, these transformations are applicable to all deductive systems that handle quantified variables schematically by using unification. This includes almost all known deductive systems, and that is why the title of this paper refers to the substitutional framework as a "general framework."

The basic idea of reasoning about taxonomic information during unification dates back to Reiter's [1977] work on deductive databases. The results reported in this paper greatly generalize his results. To begin with, his results pertain to a language that is much weeker than that employed here. His language, which can best be described as "database logic," has no existential quantifiers or function symbols other than 0-arity ones. Thus the only terms that occur in the language are constants and variables, and therefore much of the difficulty of sorted unification is avoided. Furthermore, Reiter only considered integrating sorted unification into a particular deductive system, O-resolution. He proved the completeness of sorted O-resolution, but because the proof is an argument about the syntactic form of O-deductions, there is no apparent way in which it generalizes to other forms of deduction.

Reiter also observed that there are certain cases in which sorted O-resolution is incomplete and he identified an extremely strict condition *sufficient* for its completeness. Though he pointed out that this condition is not necessary for completeness, he did not identify a necessary condition. The present paper identifies a condition that is both necessary and sufficient for completeness, not just for sorted O-resolution, but for

[1] A comparison of solutions to Schubert's Steamroller problem has been compiled by Stickel [1986]

any form of sorted deduction that falls into the substitutional framework. This result thus identifies the limitations inherent in building taxonomic reasoning into unification.

After presenting the syntax and semantics of a first-order language with sorted variables, this paper gives an overview of the substitutional approach. Section 4 presents some fundamental definitions and theorems on which the substitutional framework rests. In particular, that section generalizes the notions of substitutions and unifiers to account for sorted variables, and generalizes the Herbrand Theorem accordingly. Using resolution as an example, Section 5 illustrates how ordinary deduction systems and their completeness proofs can be transformed into sorted deduction systems and their completeness proofs.

2 The Sorted Language

This section introduces a language called Sorted First Order Predicate Calculus (SFOPC) that extends the ordinary First Order Predicate Calculus (FOPC) by introducing sort symbols and variables that are restricted to range over specified sorts. There is little about this extension that is particular to FOPC; any first-order language containing standard quantified variables can be extended in the same manner. In this paper SFOPC is used for illustrative purposes. Indeed, all the ideas, definitions and theorems presented here in terms of SFOPC were originally developed for a three-valued logic [Frisch, 1986a].

SFOPC is written with a lexicon that contains function and predicate symbols in the usual manner. In addition, the SFOPC lexicon contains a countable set of sort symbols. Typographically, sort symbols are written entirely in small capitals as such: MAMMAL. Semantically, a sort symbol, like a monadic predicate, denotes a subset of the domain, called a sort.

SFOPC consists of two disjoint sublanguages that share the same lexicon. The first sublanguage is used to express general information while the second sublanguage is for representing information about the relationships among sorts in the sort module. The sentences of the first sublanguage are called "A-sentences" while those of the second are called "S-sentences."

A-sentences are similar to ordinary FOPC sentences except that they may contain *sorted variables*, variables that are restricted to range over specified subsets of the domain. A sorted variable is a pair, $x{:}\tau$, where x is a variable name and τ, often referred to as a *restriction*, is a sort symbol. Here is an example of an A-sentence, which has a single sort symbol, DOG:

$$\forall x{:}\textsc{dog}\ \mathit{Drink}(x{:}\textsc{dog}, \mathit{beer}) \vee \mathit{Eat}(x{:}\textsc{dog}, \mathit{meat}) \quad (1)$$

To avoid confusion I never write a formula containing two distinct variables that have the same variable name. That is, no formula contains variables $x{:}\tau$ and $x{:}\tau'$ where τ and τ' are distinct. This enables use of the following shorthand. If a formula has multiple occurrences of the same variable then the restriction often is written on only the first occurrence. For example, (1) can be abbreviated as (2).

$$\forall x{:}\textsc{dog}\ \mathit{Drink}(x, \mathit{beer}) \vee \mathit{Eat}(x, \mathit{meat}) \quad (2)$$

For clarity, variables are sometimes written in angle brackets, such as $\langle x{:}\tau \rangle$. τ and ω are used as metalinguistic symbols that always stand for a restriction. Henceforth, the term "variable" refers generally to either an ordinary variable or a sorted variable. This paper's treatment of sorted variables can be generalized to the case where variables are restricted by the intersection of a finite set of sorts. For simplicity this is not done here, but see [Frisch, 1986a] for an example of this generalization.

Semantically, a sorted variable ranges only over the subset of the domain denoted by its restriction. Formally this is captured by the following semantic rules for restricted quantifiers, quantifiers with sorted variables. In these semantic rules $[\![\psi]\!]^{M,e}$ is the semantic value assigned to an expression or symbol ψ by a model M and an assignment to variables e, and $e[d/x]$ is the assignment to variables that is identical to e with the possible exception that x is assigned d. (Recall that $[\![\tau]\!]^{M,e}$ is a subset of the universe.)

$$\begin{aligned}
[\![\forall x{:}\tau\ \phi]\!]^{M,e} &= \text{True if for every } d \in [\![\tau]\!]^{M,e}, \\
&\quad\ [\![\phi]\!]^{M,e[d/x]} = \text{True} \\
&= \text{False otherwise} \\
[\![\exists x{:}\tau\ \phi]\!]^{M,e} &= \text{True if for some } d \in [\![\tau]\!]^{M,e}, \\
&\quad\ [\![\phi]\!]^{M,e[d/x]} = \text{True} \\
&= \text{False otherwise}
\end{aligned}$$

Notice that if τ is a sort symbol that denotes the entire domain in some model M, then $[\![\forall x{:}\tau\ \phi]\!]^{M,e} = [\![\forall x\ \phi]\!]^{M,e}$ and $[\![\exists x{:}\tau\ \phi]\!]^{M,e} = [\![\exists x\ \phi]\!]^{M,e}$. Consequently, unsorted variables are often treated as sorted variables implicitly restricted to the "universal" sort. Also notice that if τ denotes the empty set in some model, then that model assigns True to $\forall x{:}\tau\ \phi$ and False to $\exists x{:}\tau\ \phi$.

While A-sentences employ variables restricted by sorts, the role of S-sentences is to express relationships among the sorts and the sortal behavior of the functions. S-sentences are constructed like ordinary sentences of FOPC except that they contain no ordinary predicate symbols; in their place are sort symbols acting as monadic predicate symbols. Hence, every atomic S-formula is of the form $\tau(t)$, where τ is a sort symbol and t is an ordinary term. In the obvious way I use the terms *S-formula* and *S-literal*. Here are examples of S-sentences:

$$\textsc{dog}(\mathit{fido}) \wedge \textsc{dog}(\mathit{mother}(\mathit{fido}))$$

$$\forall x \ \text{DOG}(x) \rightarrow \text{MAMMAL}(x)$$

$$\forall x, y \ \text{ODD}(x) \wedge \text{EVEN}(x) \rightarrow \text{ODD}(sum(x, y))$$

S-formulas are assigned truth values as one would expect: an atomic formula $S(t)$ is assigned True if the domain element denoted by t is a member of the set denoted by S, and a molecular S-formula is assigned a value in the usual Tarskian manner.

A representation consisting of a set of SFOPC sentences can be partitioned into a set of A-sentences and a set of S-sentences. The S-sentences are stored in the sort module, and, according to the substitutional framework, only get called into play when the reasoning engine operating on the A-sentences performs unification. The set of S-sentences stored in the sort module will be referred to as a *sort theory*.

Instead of having a sort theory, typical sorted logics have a *sort structure*, which is an ordering on the sort symbols. The sort structure is not represented in the logical language but is used in the meta-language specification of the syntax and semantics of the language. In dealing with these sorted logics one can speak of the sort of a term. Strictly speaking, this cannot be done in SFOPC; only elements of the domain have sorts. The denotations of terms and of sorts can vary from model to model and hence a term can denote objects of different sorts in different models. Nonetheless, there are ways of translating between SFOPC and typical sorted logics but discussion of them is outside the scope of this paper.

It should be noted that SFOPC is no more expressive than FOPC; each sentence of SFOPC is logically equivalent to one (of about the same length) of FOPC. Clearly the addition of sort symbols does not make the language more expressive since they behave semantically like monadic predicate symbols. Nor does the addition of sorted variables enhance the expressiveness of the language. To see this, observe that any formula containing restricted quantifiers can be rewritten to a logically equivalent one without restricted quantifiers on the basis of two logical equivalences:

$$\forall x{:}\tau \ \psi \equiv \forall x \ \tau(x) \rightarrow \psi'$$

$$\exists x{:}\tau \ \psi \equiv \exists x \ \tau(x) \wedge \psi'$$

where ψ' is the result of substituting x for all free occurrences of $x{:}\tau$ in ψ. The formula that results from removing all restricted quantifiers from a formula ϕ by this rewriting process is called the *normalization* of ϕ and is denoted by ϕ^N. If Φ is a set of formulas, then $\Phi^N = \{\phi^N | \phi \in \Phi\}$.

To simplify the exposition of the remainder of the paper we consider only sentences in Skolem Normal Form (SNF), that is, sentences that are free of existential quantifiers and whose universal quantifiers are in prenex position (i.e., at the beginning of the sentence). Hence, "quantifier" always means universal quantifier and a quantified sentence is always a universally quantified sentence in prenex form. This assumption is common in work on deduction and does not result in a loss of generality since any set of FOPC sentences can be mechanically transformed to SNF without affecting their satisfiability or unsatisfiability. In a similar manner, any set of SFOPC sentences can be transformed to SNF [Frisch, 1986a], but the transformation is not discussed in this paper.

3 The Substitutional Framework

Since the introduction of resolution in 1965, nearly every automated deductive system has handled universally quantified variables. by using a unification algorithm. The method is ubiquitous, being used in theorem provers, rewrite systems, parsers, logic-programming systems, logic databases, question-answering systems, and planners.[2]. By paralleling this general approach to deduction with ordinary variables, the substitutional framework for deduction with sorted variables achieves the same degree of generality.

The gist of this approach is that a quantified sentence is treated as a schema standing for the set of its ground instances. The justification for this is found in Herbrand's Theorem, which states that a set of quantified sentences is satisfiable if, and only if, the set containing every ground instance of every one of the sentences is satisfiable.

A deductive system that operates on ground sentences can be made to operate on quantified sentences by replacing tests for equality between expressions with tests for unifiability between them. Sometimes this is all that needs to be done, though sometimes additional mechanisms must be incorporated. Regardless of which is the case, the idea is that deductions on quantified sentences are themselves schematic for deductions on ground sentences. A schematic deduction is said to *lift* each deduction that is an instance of it.

Given a deductive system based on unification we would like to show that it does indeed treat quantified sentences as schemas. This usually takes the form of a *lifting theorem* for the deductive system stating that every deduction that can be made from the ground instances of a set of sentences can be made schematically from the sentences themselves.

The substitutional approach to handling restricted quantifiers is built upon the notion of *well-sorted substitution* in the same way that the method for handling ordinary quantifiers is built upon the notion of substitution. The well-sortedness of a substitution is relative to a sort theory; thus to simplify terminology a substitution that is well-sorted with respect to sort theory Σ is called a Σ-*substitution*. Intuitively, a Σ-substitution is a substitution that respects the restrictions attached to the variables it replaces. Thus, an algorithm for

[2]The substitutional framework applies to all of these systems. Collectively, these systems are referred to as "deductive systems."

performing Σ-unification must query the sort module in order to obtain certain information about Σ. The sort module must deduce an answer to the query from the sort theory. The computational complexity of responding to these queries, and hence the complexity of Σ-unification, depends upon the structure of Σ.

It is a simple matter to extend an unsorted deductive system $\mathcal{D}$ to a sorted deductive system, $\mathcal{D}_S$, that operates on the new, sorted language. All that need be done is to replace the unification algorithm of $\mathcal{D}$ with a sorted unification algorithm. Notice that the resulting system, $\mathcal{D}_S$, uses the sort module in only one place—in its computation of unifiers. Thus, the interface between the original deductive methods of $\mathcal{D}$ and the newly-incorporated deductive methods of the sort module is extremely simple. Furthermore, integrating the two kinds of deductive methods does not involve restructuring $\mathcal{D}$. So, for example, if $\mathcal{D}$ is based on a set of proof rules, no new rules need be added to obtain the corresponding proof system for the sorted language. On the other hand, in extending a resolution proof system to deal with sorted logic, Cohn [1983] was forced to formulate and incorporate additional proof rules; the expressiveness of his logic makes it impossible to build a complete deductive system by modifying only the unification component of a standard resolution system.

$\mathcal{D}_S$ treats quantified sentences as sentence schemas in much the same way that $\mathcal{D}$ does. The only difference is that a sentence with sorted variables does not stand for the set of all its ground instances, but only for its well-sorted ground instances. This treatment of restricted quantifiers is justified by the *Sorted Herbrand Theorem*, according to which a sort theory Σ and a set S of sentences with restricted quantifiers are jointly satisfiable if, and only if, the set containing every Σ-ground instance of every sentence in S is satisfiable. In saying that S and Σ can be replaced by the Σ-ground instances of S, the Sorted Herbrand Theorem justifies using Σ only in computing Σ-unifiers. Notice that Σ must be used in generating the correct instances of S, but once they are obtained Σ is irrelevant because ground instances have no variables and hence no sort symbols.

To show that $\mathcal{D}_S$ treats schemas with sorted variables properly one must prove the *Sorted Lifting Theorem* for $\mathcal{D}_S$. This theorem asserts that every deduction that can be made from the Σ-ground instances of a set of sorted sentences can be made schematically from the sentences themselves. Section 5 presents a simple, systematic way of taking a proof of the Lifting Theorem for any $\mathcal{D}$ and producing a proof of the Sorted Lifting Theorem for $\mathcal{D}_S$.

4 Fundamental Definitions and Results

4.1 Well Sorted Substitutions and Unifiers

Roughly speaking, a substitution is well sorted relative to a sort theory if it maps each variable to a term that satisfies the restriction associated with the variable. More precisely, a substitution θ is well sorted relative to a sort theory Σ if, and only if, for every variable $x{:}\tau$, $\langle x{:}\tau\rangle\theta$ is a term t such that $\Sigma \models \forall\, \tau(t)$. Here, and in general, an expression of the form $\forall\, \phi$ denotes the universal closure of ϕ—that is, the formula $\forall x_1 \cdots x_n\ \phi$ where $x_1, \ldots, x_n$ are the freely-occurring variables of ϕ.

Two special cases of this definition are worth noting. If θ is well sorted relative to Σ and maps $x{:}\tau$ to a ground term t, then it must be that $\Sigma \models \tau(t)$. In other words, Σ must entail that t is of sort τ. If θ maps $x{:}\tau$ to a variable $y{:}\omega$ then it must be that $\Sigma \models \forall y{:}\omega\ \tau(y)$. That is, Σ must entail that ω is a subset of τ.

Expression e' is said to be a well sorted instance of e relative to Σ if $e' = e\theta$, for some substitution θ that is well sorted relative to Σ. In the obvious way, I speak of well sorted ground instances of a formula and write $e_{\Sigma gr}$ to denote the set of all ground instances of e that are well sorted relative to Σ.

Substitutions, well-sorted or not, are functions and therefore can be composed. If θ and σ are substitutions then their composition, $\theta \cdot \sigma$, is $\lambda e.\sigma(\theta(e))$. In other words, $\theta \cdot \sigma$ is such that $e(\theta \cdot \sigma) = (e\theta)\sigma$.

The usual notions of what it means for a substitution to be a unifier and for one substitution to be more general than another can be adapted to well sorted substitutions as follows:

Definition 1 (Well Sorted Unifier) *Let E be a set of expressions and θ be a substitution. θ is a* well sorted unifier *of E relative to Σ if it is a unifier of E and is well sorted relative to Σ.*

Definition 2 (Σ-More General) *Let θ_1 and θ_2 be substitutions that are well sorted relative to Σ. θ_1 is* Σ-more general *than θ_2 (written $\theta_1 \geq_\Sigma \theta_2$) iff $\theta_1 \cdot \sigma = \theta_2$ for some substitution σ that is well sorted relative to Σ.*

The set of all ordinary unifiers of a set of expressions contains unifiers that are more general than every other unifier in the set. These so-called most general unifiers are important because any one of them can serve as a representative for the entire set. On the other hand, the set of all well-sorted unifiers of a set of expressions may not contain a most general unifier. It will, however, contain *maximally general unifiers*, unifiers such that no other unifiers are strictly more general. Here is an example:

Example 3 (Σ-unifiers) *Let*

$$\Sigma = \{\forall x, y\ \text{ODD}(x) \wedge \text{ODD}(y) \rightarrow \text{EVN}(sum(x,y)),$$
$$\forall x, y\ \text{EVN}(x) \wedge \text{EVN}(y) \rightarrow \text{EVN}(sum(x,y))\}$$
$$E = \{z{:}\text{EVN}, sum(v,w)\}$$
$$\theta_1 = \{sum(x{:}\text{EVN}, y{:}\text{EVN})/z, x{:}\text{EVN}/v, y{:}\text{EVN}/w\}$$
$$\theta_2 = \{sum(x{:}\text{ODD}, y{:}\text{ODD})/z, x{:}\text{ODD}/v, y{:}\text{ODD}/w\}$$

Then θ_1 and θ_2 are each maximally-general Σ-unifiers of E and neither θ_1 nor θ_2 is Σ-more general than the other.

Indeed, a finite set of expressions may have an infinite number of maximally-general unifiers, none of which are more general than any others. For instance:

Example 4 (Σ-unifiers) *Let*

$$\Sigma = \{\forall x\ \text{T}(i(x)) \rightarrow \text{T}(i(s(x))), \text{T}(i(a))\}$$
$$E = \{z{:}\text{T}, i(s(y))\}$$

Then

$$\{\{a/y, i(s(a))/z{:}\text{T}\}, \{s(a)/y, i(s(s(a)))/z{:}\text{T}\}, \ldots\}$$

is an infinite set of maximally general Σ-unifiers of E and none of these is Σ-more general than any of the others.

In performing unsorted deduction there is no need to consider all unifiers of two expressions; it suffices to consider only a most general unifier. A most general unifier forms a basis from which all other unifiers can be generated by composing it with other substitutions. A corresponding basis also can be formed for the set of unifiers of two sorted expressions. As the above examples suggests, it may be necessary to form the basis from a set of unifiers rather than a single unifier. Since the basis should be a small as possible it should not contain two substitutions such that one is more general than the other. The basis we speak of is called a Σ-*most general complete set of unifiers* (or Σ*MGCU*) and is defined as follows.

Definition 5 (Σ-MGCU) *Let E be a set of expressions and let Θ be a set of substitutions that are well sorted relative to Σ. Then Θ is a* Σ-most general complete set of unifiers *(or Σ*MGCU*) of E iff:*

1. *Θ is correct; if $\theta \in \Theta$ and $\theta \geq_\Sigma \theta'$ then θ' is a Σ-unifier of E.*
2. *Θ is complete; if θ' is a Σ-unifier of E then for some $\theta \in \Theta, \theta \geq_\Sigma \theta'$.*
3. *Θ is most general; Θ does not contain two distinct substitutions such that one is Σ-more general than the other.*

It is easy to verify that every element of a Σ*MGCU* is a maximally-general Σ-unifier. Suppose that θ is a non-maximal Σ-unifier and is in the set. Then there is some unifier θ_1 that is strictly Σ-more general than θ and since the set is complete it contains some substitution $\theta_2 \geq_\Sigma \theta_1$. But then the set would contain comparable elements, θ_2 and θ, thus contradicting the third condition of the definition of Σ*MGCU*.

There is no effective procedure for finding a Σ*MGCU* of two given sorted-expressions relative to an arbitrary sort theory. However there are restrictions that can be placed on the sort theory so that such procedures do exist. Space precludes an extensive discussion of sorted unification and algorithms for computing it. However, since the substitutional framework depends on the decidability of sorted unification, Appendix A presents a particular restriction on the form of a sort theory, the monomorphic tree restriction, and gives an algorithm that computes a Σ*MGCU* of any two expressions provided that Σ meets the restriction.

Because they are substitutions, well sorted substitutions enjoy all the properties possessed by substitutions in general. So, for example, since the composition of substitutions is associative, so is the composition of well sorted substitutions. For other reasons well sorted substitutions have many of the other properties that ordinary substitutions do, such as those stated in the following lemmas.

Lemma 6 (Identity Lemmas) *The identity substitution, ϵ, is well sorted relative to any sort theory.*

Proof: ϵ maps every variable $x{:}\tau$ to itself and any sort theory entails $\forall x{:}\tau\ \tau(x)$ since $\forall x{:}\tau\ \tau(x)$ is a valid sentence. □

Lemma 7 (Composition Lemma) *If σ and θ are well sorted substitutions relative to Σ, then so is $\theta \cdot \sigma$.*

Proof: I assume θ and σ are Σ-well sorted, and show that for any variable, $x{:}\tau$, $\Sigma \models \forall\ \tau(\langle x{:}\tau\rangle\theta\sigma)$. Let $\phi[y_1{:}\omega_1, \ldots, y_n{:}\omega_n]$ be $\langle x{:}\tau\rangle\theta$, where $y_1{:}\omega_1, \ldots, y_n{:}\omega_n$ are the free variables of the term. (Subsequently, an expression of the form $\psi[t_1, \ldots, t_n]$ shall refer to the expression that results from replacing all free occurrences of $y_i{:}\omega_i$ in ϕ by t_i, for $1 \leq i \leq n$.) Since θ is Σ-well sorted, Σ entails

$$\forall\ \tau(\phi[y_1{:}\omega_1, \ldots, y_n{:}\omega_n])$$

which normalized is

$$\forall\ \omega_1(y_1) \wedge \cdots \wedge \omega_n(y_n) \rightarrow \tau(\phi[y_1, \ldots, y_n]) \quad (3)$$

For $1 \leq i \leq n$ let ψ_i be $\langle y_i{:}\omega_i\rangle\sigma$. Since σ is Σ-well sorted, Σ entails

$$\forall\ \omega_1(\psi_1) \wedge \cdots \wedge \omega_n(\psi_n) \quad (4)$$

By combining (4) and (3) with modus ponens we can conclude that Σ entails

$$\forall\ \tau(\phi[\psi_1, \ldots, \psi_n])$$

which is

$$\forall\ \tau(\langle x{:}\tau\rangle\theta\sigma)$$

□

4.2 The Sorted Herbrand Theorem

The statement of the Sorted Herbrand Theorem relies upon the commonly-used notion of a least Herbrand model.[3] The Herbrand models can be partially ordered in the following way:

> $M_1 \leq M_2$ iff for every predicate symbol or sort symbol P, $[P]^{M_1} \subseteq [P]^{M_2}$

A set of sentences has a least Herbrand model if one of its Herbrand models is less than all its others.

Theorem 8 (Sorted Herbrand Theorem) [4] *Let α be a set of A-clauses and let Σ be a set of S-clauses that has a least Herbrand model. Then $\alpha \cup \Sigma$ is unsatisfiable if, and only if, $\alpha_{\Sigma gr}$ is.*

Proof:
$\Longleftarrow$ $\alpha \cup \Sigma$ entails each element of $\alpha_{\Sigma gr}$. Thus, if $\alpha_{\Sigma gr}$ is unsatisfiable then so must be $\alpha \cup \Sigma$.
$\Longrightarrow$ If $\alpha_{\Sigma gr}$ is satisfiable then it is satisfied by a Herbrand model. Indeed, since $\alpha_{\Sigma gr}$ contains no sort symbols, it is satisfied by a Herbrand model M that assigns every sort symbol the same denotation that is assigned by the least Herbrand model of Σ. I show that M also satisfies $\Sigma \cup (akb^N)_{gr}$. Clearly M satisfies Σ. I now consider a, an arbitrary sentence in α and show that M satisfies $(a^N)\theta$, an arbitrary ground instance of a^N. $(a^N)\theta$ is of the form $T_1 \wedge \cdots \wedge T_n \rightarrow \beta$. If $\beta \notin \alpha_{\Sigma gr}$ then $\Sigma \not\models T_1 \wedge \cdots \wedge T_n$. Since $T_1 \wedge \cdots \wedge T_n$ is satisfied by some model of Σ it must be satisfied by M, which agrees with the least Herbrand model of Σ. On the other hand, if $\beta \in \alpha_{\Sigma gr}$ then M satisfies β (since it satisfies $\alpha_{\Sigma gr}$) and therefore also satisfies $(a^N)\theta$. □

What happens if Σ has more than one minimal Herbrand model? Consider the case where

$$\begin{aligned} \Sigma &= \{\textsc{baby}(\mathit{Ralph}) \vee \textsc{dog}(\mathit{Ralph})\} \\ \alpha &= \{\forall x{:}\textsc{dog}\ \mathit{Annoys}(x, \mathit{Alan}), \\ &\quad \forall x{:}\textsc{baby}\ \mathit{Annoys}(x, \mathit{Alan})\} \end{aligned}$$

Observe that Σ has two minimal Herbrand models; one that satisfies only $\textsc{baby}(\mathit{Ralph})$ and another that satisfies only $\textsc{dog}(\mathit{Ralph})$. Because Σ does not entail any atomic sentences, $\alpha_{\Sigma gr}$ is empty. Now, here is the problem: $\Sigma \cup \alpha \models \mathit{Annoys}(\mathit{Ralph}, \mathit{Alan})$, but $\alpha_{\Sigma gr}$ does not.

Reiter [1977] noticed that this difficulty arose in his work on deductive databases. His solution was to insist that Σ satisfies a condition called "τ-completeness"—that for every sort symbol ω and every term t either $\Sigma \models \omega(t)$ or $\Sigma \models \neg\omega(t)$. This condition is equivalent to requiring that Σ has a unique Herbrand model, not merely a unique *minimal* one. Though Reiter found a sufficient condition, it is grossly over-restrictive. What about the condition that Σ must have a least Herbrand model? Is it also over-restrictive? The Necessity/Sufficiency Theorem asserts that the condition is indeed necessary. The proof of the theorem demonstrates that for any sort theory having multiple minimal Herbrand models an example like the above baby-and-dog one can be constructed.

Lemma 9 (Least Model Lemma) *A satisfiable set of clauses Ψ has a least Herbrand model iff for any finite disjunction of ground atomic formulas, $A = \psi_1 \vee \cdots \vee \psi_n$, $\Psi \models A$ implies $\Psi \models \psi_i$, for some $1 \leq i \leq n$.*

Proof:
$\Longleftarrow$ Let M be the greatest lower bound of all Herbrand models of Ψ. Assuming the antecedent of the theorem, I show that M satisfies Ψ and, thus, is the least Herbrand model of Ψ. Let $C = \neg\alpha_1 \vee \cdots \vee \neg\alpha_k \vee \beta_1 \vee \cdots \vee \beta_m$ be an arbitrary clause in Ψ_{gr}. If some model of Ψ satisfies one $\neg\alpha_i$ then so does M and hence M satisfies C. Otherwise every model of Ψ falsifies every $\neg\alpha_i$ and hence $\Psi \models \beta_1 \vee \cdots \vee \beta_m$. By the assumption, there is an i such that $\Psi \models \beta_i$. Thus, M satisfies β_i and therefore it also satisfies C. Either way, M satisfies C, an arbitrary member of Ψ_{gr}, and therefore M satisfies Ψ.
$\Longrightarrow$ Assume Ψ has a least Herbrand model M and that $A = \psi_1 \vee \cdots \vee \psi_n$ is a disjunction of ground atomic sentences such that $\Psi \models A$. Then it must be that M satisfies A, and hence satisfies some ψ_i. But since M is a least model, every Herbrand model of Ψ satisfies α. Therefore $\Psi \models \psi_i$. □

Theorem 10 (Necessity/Sufficiency Theorem) *Let Σ be a set of S-clauses. It is both* necessary and sufficient *that Σ has a least Herbrand model for the following statement to hold:*

> *For every set of A-clauses α, $\Sigma \cup \alpha$ is unsatisfiable if, and only if, $\alpha_{\Sigma gr}$ is.*

Proof:
Sufficiency: This is equivalent to the Sorted Herbrand Theorem.
Necessity: I assume that Σ does not have a least Herbrand sort assignment and construct a set of A-clauses α such that $\alpha \cup \Sigma$ is unsatisfiable but $\alpha_{\Sigma gr}$ is satisfiable. By the Least Model Lemma, there is a disjunction of ground atomic S-formulas, $\psi = \mathrm{P}_1(t_1) \vee \cdots \vee \mathrm{P}_n(t_n)$ such that $\Sigma \models \psi$ and for every $1 \leq i \leq n$, $\Sigma \not\models \mathrm{P}_i(t_i)$. Let α be the set of sentences

$$\{\forall x{:}\mathrm{P}_i\ Q_i(x) \mid 1 \leq i \leq n\} \cup \{\neg Q_i(t_i) \mid 1 \leq i \leq n\}$$

It is now easy to verify that $\alpha_{\Sigma gr}$ is satisfiable even though $\alpha \cup \Sigma$ is not. □

[3] These least Herbrand Herbrand models are central to the theory of logic programming. In that literature, least models are often called *unique minimal models.*

[4] For simplicity this theorem is stated in terms of clauses. Nonetheless, like the unsorted Herbrand Theorem, it applies to sentences that are in Skolem Normal Form.

5 Example: Sorted Resolution

This section uses standard resolution to illustrate how the substitutional framework can be used to systematically extend an unsorted deductive system and its proof of completeness into a sorted deductive system and its proof of completeness.

The resolution rule of inference operates on clauses, each of which we shall take to be a set of literals. If L is a set of literals, then $\bar{L}$ denotes the set containing the complement of every literal in L. Assume that L and M are two FOPC clauses whose variables have been standardized apart. If some $l \subseteq L$ and $m \subseteq M$ are such that $l \cup \bar{m}$ is unifiable by a maximally general unifier θ, then the clause $(L - l)\theta \cup (M - m)\theta$ is a resolvent of L and M. A resolution derivation of FOPC clause C from a set of FOPC clauses S is a binary tree such that its root is C, each leaf is a member of S, and each interior node is a resolvent of its daughter nodes. We shall write $S \vdash_{\overline{\text{RES}}} C$ to assert that there is a resolution derivation of C from S.

It is now straightforward to extend the definition of resolvent to the SFOPC case where the two clauses may contain sorted variables and the resolvent is relative to a set of S-sentences Σ. Such a resolvent is the same as an ordinary resolvent except that the substitution involved, θ, must be a maximally general Σ-unifier. Let us make this explicit by considering a set of S-sentences Σ and two A-clauses of SFOPC, L and M, whose variables have been standardized apart. If some $l \subseteq L$ and $m \subseteq M$ are such that $l \cup \bar{m}$ are unifiable by a maximally general Σ-unifier θ, then the clause $(L - l)\theta \cup (M - m)\theta$ is a Σ-resolvent of L and M. The definition of resolution derivation extends in the obvious straightforward manner. A Σ-resolution derivation of A-clause C from a set of A-clauses S is a binary tree such that its root is C, each of its leaves is a member of S, and each of its interior nodes is a Σ-resolvent of its daughter nodes. We shall write $S \vdash_{\overline{\Sigma\text{RES}}} C$ to assert that there is a Σ-resolution derivation of C from S.

In unsorted resolution, given two clauses and a particular set of literals to be resolved upon, one need only consider a single resolvent. All other resolvents are variants of it. However, in sorted resolution one may need to consider many resolvents—one for each substitution in a $\Sigma MGCU$ of the literals to be resolved upon. All other resolvents are variants of these.

The Completeness Theorem for Resolution states that a set S of FOPC clauses is unsatisfiable only if $S \vdash_{\overline{\text{RES}}} \square$. Typically, proofs of this theorem take the following form. First one proves the completeness of resolution for the ground case and then one connects this up to the non-ground case. Assuming S is unsatisfiable, the Herbrand Theorem for FOPC tells us that S_{gr} also is unsatisfiable, from which the completeness of ground resolution tells us that $S_{gr} \vdash_{\overline{\text{RES}}} \square$. From this, a theorem known as the Lifting Theorem for Resolution tells us that $S \vdash_{\overline{\text{RES}}} \square$.

Many first-order deductive systems that handle variables schematically via unification can be proved complete in this manner. One proves that the system is complete in the ground case and that ground proofs can be lifted, and then couples this with the Herbrand Theorem.

Now consider proving the completeness of sorted resolution.

Theorem 11 (Completeness of Σ-Resolution) *Let Σ be a set of S-clauses with a least Herbrand model and α be a set of A-clauses. If $\Sigma \cup \alpha$ is unsatisfiable then $\alpha \vdash_{\overline{\Sigma\text{RES}}} \square$.*

The proof of this theorem parallels that for the completeness of unsorted resolution. Assuming $\Sigma \cup S$ is unsatisfiable, the Sorted Herbrand Theorem tells us the $S_{\Sigma gr}$ is also unsatisfiable. Then the completeness theorem for ground resolution tells us that $S_{\Sigma gr} \vdash_{\overline{\Sigma\text{RES}}} \square$. We know that Σ-resolution is complete for ground clauses because it is the same as ordinary resolution on ground clauses. To complete the proof one must resort to the Lifting Theorem for Σ-resolution, which tells us that if $S_{\Sigma gr} \vdash_{\overline{\Sigma\text{RES}}} \square$ then $S \vdash_{\overline{\Sigma\text{RES}}} \square$.

So, a completeness proof of the form used for ordinary resolution can be modified systematically to a completeness proof for the corresponding sorted deduction system. All references to the Herbrand Theorem are replaced with references to the Sorted Herbrand Theorem. The ground completeness theorem remains unchanged. All that needs to be done is that a lifting theorem for the particular sorted deduction system must be proved. As we will now see, this too can be done in a systematic manner.

For any set S of clauses and any clause C the Lifting Theorem for Resolution says that $S \vdash_{\overline{\text{RES}}} C$ if $S_{gr} \vdash_{\overline{\text{RES}}} C'$ for some $C' \in C_{gr}$. The corresponding theorem for sorted resolution follows.

Theorem 12 (Σ-Resolution Lifting) *Let S be a set of A-clauses, let C be an A-clause and let Σ be a set of S-sentences. Then, $S \vdash_{\overline{\Sigma\text{RES}}} C$ if $S_{\Sigma gr} \vdash_{\overline{\Sigma\text{RES}}} C'$ for some $C' \in C_{gr}$.*

A proof of the Sorted Lifting Theorem can be produced by systematically modifying the proof of the Lifting Theorem. Simply replace all occurrences of the words "substitution," "unifier" and "maximally general unifier" with the words "Σ-substitution," "Σ-unifier" and "Σ-maximally general unifier" respectively. The resulting argument is correct because all properties of substitutions, unifiers, and maximally general unifiers on which the original proof relies are also properties (as established in Section 4.1) of Σ-substitutions, Σ-unifiers, and Σ-maximally general unifiers.

6 Conclusions

Though this paper has left unanswered many questions regarding sorted deduction, it has proposed a frame-

work under which such questions can be addressed for an entire class of deduction systems rather than for one system at a time.

As this paper presents sorted deduction in the context of sorted first-order logic, sorted variables are introduced by restricted quantifiers. These sorted quantified variables can be treated as sorted schematic variables on the basis of the Sorted Herbrand Theorem. Nonetheless, the substitutional framework can be applied to systems that introduce schematic variables directly. For example, there is no reason why production systems or grammar systems that use unsorted variables cannot be extended to incorporate sorted variables. The results presented here concerning sorted substitution, sorted unification, and sorted lifting are as applicable in those contexts as in the logic context.

Appendix A: The Monomorphic Tree Restriction and Sorted Unification

Here we present both a particularly severe restriction on the form of a sort theory and an algorithm, which is shown in Figure 1, that computes a $\Sigma MGCU$ of any two expressions provided that Σ meets the restriction. Further results on sorted unification have been presented by Walther [1988] and by Schmidt-Schauss [1985].

The restriction on the sort theory is the combination of two restrictions, the tree restriction and the monomorphism restriction, and hence is called the *monomorphic tree restriction.* A sort theory meeting the tree restriction consists only of sentences of two kinds. Sentences of the first kind are of the form $\forall x\ \tau(x) \rightarrow \tau'(x)$, where τ and τ' are sort symbols. Furthermore, the tree restriction requires that for any τ there is at most one sentence of the above form. Sentences of the second kind are of the form $\forall\ \tau(t)$ where τ is a sort symbol and t is a term. Furthermore, the tree restriction requires that the sort theory does not contains two sentences, $\forall\ \tau(t)$ and $\forall\ \tau'(t')$, such that $\tau(t)$ and $\tau'(t')$ are unifiable.

The tree restriction affects what sentences logically follow from a sort theory. The effects can be easily described using the notion of the *extension* of a sort symbol. The extension of τ is the set of all terms t such that Σ entails $\forall\ \tau(t)$. The effect of the tree restriction can now be stated easily: if the extensions of two sort symbols have a non-empty intersection, then one of the extensions is a subset of the other.

The monomorphism restriction imposes on every sentence of the form $\forall\ \tau(t)$ the additional restriction that t must be of the form $f(x_1, ..., x_n)$ where f is an n-ary function symbol for some $n \geq 0$. The effect of this restriction is that a non-variable term is in the extension of a sort symbol if, and only if, every instance of the term is in the extension. Consequently, one can determine whether a non-variable term is in the extension of a sort symbol solely on the basis of the term's main function symbol.

If a sort theory Σ meets the monomorphic tree restriction then any two Σ-unifiable expressions have a Σ-most general unifier; in other words, they have a singleton $\Sigma MGCU$. Notice that in both Example 3 and in Example 4 Σ violates the tree restriction.

Given a two expressions and a sort theory Σ meeting the monomorphic tree restriction the algorithm in Figure 1, whose form is based on Martelli and Montanari's [1982] algorithm for ordinary unification, determines whether the two expressions are Σ-unifiable, and if so it returns a Σ-most general unifier. The algorithm assumes that we have a procedure for determining whether the extension of one sort symbol, τ, is a subset of the extension of another, τ', (i.e., whether $\Sigma \models \forall x\ \tau(x) \rightarrow \tau'(x)$) and one for determining whether a term t is in the extension of a sort symbol τ (i.e., whether $\Sigma \models \forall\ \tau(t)$). Both of these can be determined easily by SLD-resolution.

Like the Martelli and Montanari algorithm, this algorithm operates by repeatedly transforming a set of equations between expressions to be unified until no more transformations apply, whereupon the most general unifier can be read directly from the equations. The unsorted unification algorithm uses four transformations, the first three of which are used unadultered in the sorted algorithm. The fourth unsorted transformation deals with equations in which the left side is a variable and the right is any term. In the sorted algorithm this transformation is divided in two; transformation 4 handles those cases in which the right side is not a variable while transformation 5 handles the cases in which it is. Each of these two transformations first checks the well-sortedness of the unifier it is building and, if this succeeds, then proceeds as in the unsorted case. Transformation 4 checks that a term that gets substituted for a variable is in the extension of the sort of the variable. When transformation 5 unifies two variables, $x{:}\tau$ and $x'{:}\tau'$, it must check the extensions of τ and τ'. If the extension of τ' is a subset of the extension of τ then $x'{:}\tau'$ is substituted for $x{:}\tau$. If the extension of τ is a subset of the extension of τ' then $x{:}\tau$ is substituted for $x'{:}\tau'$. If neither extension is a subset of the other then the unification fails.

The final difference between the unsorted and sorted algorithm is that the sorted one must check every equation for well-sortedness. If the unsorted algorithm starts with the single equation $x = f(a)$ then it can halt immediately and report that $\{f(a)/x\}$ is a most general unifier. However, the sorted algorithm should only report a unifier if $f(a)$ is in the extension of the sort of x. Thus whenever an equation is checked for well-sortedness it is marked and the algorithm does not report a successful unification until all equations have been marked.

Input: Two expressions, s_1 and s_2, and a sort theory Σ meeting the monomorphic tree restriction.
Output: SUCCESS or FAILURE; if SUCCESS then a substitution is also output.

Let X be the singleton set of equations $\{s_1 = s_2\}$
Repeatedly perform any of the following transformations until no transformation applies:

1. Select any equation of the form $t = x{:}\tau$ where t is not a variable and $x{:}\tau$ is a variable, and rewrite it as $x{:}\tau = t$.
2. Select any equation of the form $x{:}\tau = x{:}\tau$ where $x{:}\tau$ is a variable and erase it.
3. Select any equation of the form $t = t'$ where neither t nor t' is a variable.
   If t and t' are atomic and identical then erase the equation
   Else if t and t' are atomic and different then FAIL
   Else if t is composed of $e_1, \ldots, e_n$ and t' is composed of $e'_1, \ldots, e'_n$ in the same manner
   then replace $t = t'$ by the equations $e_1 = e'_1, \ldots, e_n = e'_n$
   Else FAIL.
4. Select any unmarked equation of the form $x{:}\tau = t$ where $x{:}\tau$ is a variable and t is not
   If $x{:}\tau$ occurs in t then FAIL
   Else if $\Sigma \models \forall\, \tau(t)$
   then apply the substitution $\{t/x{:}\tau\}$ to all other equations (without erasing $x{:}\tau = t$)
   mark $x{:}\tau = t$ "reduced"
   Else FAIL
5. Select any unmarked equation of the form $x{:}\tau = x'{:}\tau'$ where $x{:}\tau$ and $x'{:}\tau'$ are distinct variables
   If $\Sigma \models \forall x\ \tau'(x) \rightarrow \tau(x)$
   Then apply the substitution $\{x'{:}\tau'/x{:}\tau\}$ to all other equations (without erasing $x{:}\tau = x'{:}\tau'$)
   mark $x{:}\tau = x'{:}\tau'$ "reduced"
   Else if $\Sigma \models \forall x\ \tau(x) \rightarrow \tau'(x)$
   then replace $x{:}\tau = x'{:}\tau'$ with $x'{:}\tau' = x{:}\tau$
   apply the substitution $\{x{:}\tau/x'{:}\tau'\}$ to all other equations (without erasing $x'{:}\tau' = x{:}\tau$)
   mark $x'{:}\tau' = x{:}\tau$ "reduced"
   Else FAIL

SUCCEED with $\{t_1/x_1, \ldots, t_n/x_n\}$ where $x_1 = t_1, \ldots, x_n = t_n$ are the equations that remain in X.

Figure 1: Sorted Unification Algorithm

References

[Aït-Kaci and Nasr, 1985] Hassan Aït-Kaci and Robert Nasr. *LOGIN: A Logic Programming Language with Built-in Inheritance.* MCC Technical Report Number AI-068-85, Microelectronics and Computer Technology Corporation, July 1985.

[Allen, 1984] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, July 1984.

[Brachman *et al.*, 1983] Ronald J. Brachman, Richard E. Fikes, and Hector J. Levesque. KRYPTON: Integrating terminology and assertion. In *Proceedings AAAI-83*, pages 31–35, Washington, D. C., American Association for Artificial Intelligence, August 1983.

[Cohn, 1983] Anthony G. Cohn. *Mechanising a Particularly Expressive Many Sorted Logic.* PhD thesis, Department of Computer Science, University of Essex, 1983.

[Cohn, 1985] Anthony G. Cohn. On the solution of *Schubert's Steamroller* in many sorted logic. In *Proceedings IJCAI-85*, pages 1169–11174, Los Angeles, California, August 1985.

[Cohn, 1987] Anthony G. Cohn. A more expressive formulation of many sorted logic. *Journal of Automated Reasoning*, 3(2):113–200, 1987.

[Feferman, 1974] S. Feferman. Applications of many sorted interpolation theorems. In *Proceedings of the Symposium on Pure Mathematics*, pages 102–148, American Mathematical Society, 1974.

[Frisch, 1986a] Alan M. Frisch. *Knowledge Retrieval as Specialized Inference.* PhD thesis, Computer Science Department, University of Rochester, August 1986.

[Frisch, 1986b] Alan M. Frisch. Parsing with restricted quantification: An initial demonstration. *Computational Intelligence*, 2(3):142–150, 1986.

[Frisch *et al.*, 1983] Alan M. Frisch, James F. Allen, and Mark Giuliano. An overview of the HORNE logic programming system. *SIGART Newsletter*, (84):27–29, 1983.

[Hayes, 1971] Patrick J. Hayes. A logic of actions. In Bernard Meltzer and Donald Michie, editors, *Machine Intelligence 6*, pages 495–520, Edinburgh University Press, 1971.

[Hayes, 1985] Patrick J. Hayes. Naive physics 1: ontology for liquids. In Jerry R. Hobbs and Robert C. Moore, editors, *Formal Theories of the Commonsense World*, chapter 3, pages 71–107, Ablex Publishing Corporation, Norwood, NJ, 1985.

[Martelli and Montanari, 1982] Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, April 1982.

[McDermott, 1982] Drew McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6(2):101–155, 1982.

[McSkimin and Minker, 1979] James R. McSkimin and Jack Minker. A predicate calculus based semantic network for deductive searching. In Nicholas V. Findler, editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 205–238, Academic Press, New York, 1979.

[Patel-Schneider, 1987] Peter F. Patel-Schneider. A hybrid, decidable, logic-based knowledge representation system. *Computational Intelligence*, 3(2):64–77, May 1987.

[Reiter, 1977] Raymond Reiter. *An Approach to Deductive Question-answering.* BBN Technical Report 3649, Bolt Beranek and Newman, Inc., 1977.

[Schmidt-Schauss, 1985] M. Schmidt-Schauss. Unification in many-sorted calculus with declarations. In *Ninth German Workshop on Artificial Intelligence*, pages 118–132, Dassel/Solling, 1985.

[Stickel, 1985] Mark E. Stickel. Automated deduction by theory resolution. In *Proceedings IJCAI-85*, pages 455–458, Los Angeles, California, August 1985. An expanded version appears as Technical Note 340, Artificial Intelligence Center, SRI International, October 1984.

[Stickel, 1986] Mark E. Stickel. Schubert's Steamroller problem: Formulations and solutions. *Journal of Automated Reasoning*, 2(1):89–101, 1986.

[Vilain, 1985] Marc Vilain. The restricted language architecture of a hybrid representation system. In *Proceedings IJCAI-85*, pages 547–551, Los Angeles, California, August 1985.

[Walther, 1985] Christoph Walther. A mechanical solution of Schubert's Steamroller by many-sorted resolution. *Artificial Intelligence*, 26(2):217–224, 1985.

[Walther, 1987] Christoph Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation.* Morgan Kaufman, Los Altos, CA, 1987.

[Walther, 1988] Chtistoph Walther. Many-sorted unification. *Communications of the ACM*, 35(1):1–17, January 1988.

Default Reasoning, Minimality and Coherence

Hector Geffner*
Cognitive Systems Lab.
Computer Science Department
UCLA
Los Angeles, CA 90024

Abstract

A preferential semantics for default reasoning is presented. A partial order is defined over *classes* of models which establishes a preference for classes with a minimal set of *unexplained* exceptions. Exceptions are explained in terms of justifications which are syntactically extracted from the knowledge base. The resulting semantics succeeds in pruning the spurious models which arise in minimal model semantics, legitimizing a behavior in closer correspondence with intuition. Likewise, the proposed framework unifies and extends ideas stemming from work in default reasoning, logic programming and abductive reasoning.

1 Introduction

In [McCarthy, 80;86] McCarthy proposed circumscription as a simple but powerful second order axiom capable of endowing first order logic with non-monotonic features. In model theoretic terms, circumscription can be understood as replacing the traditional notion of entailment as truth in all models by a weaker, defeasible form of entailment in which only a subset of minimal models is considered. [McCarthy, 80; Lifschitz, 85; Etherington, 88].

Since then, several studies have analyzed the mathematical properties of circumscription.[1] Less attention, however, has been given to the circumscriptive framework as a framework for representing commonsense knowledge. In this regard, recent work has illustrated that, more often than not, the inferences sanctioned by circumscription from relatively simple conceptualizations turn out to be weaker than expected (e.g. [Hanks and McDermott, 86; Haugh, 88]). Minimal, unintended models often arise which prevent certain intended conclusions from being certified.

*This work was supported in part by National Science Foundation Grant # IRI-8610155.

[1] See [Reiter, 87] for a relevant bibliography.

This mismatch between intended meaning and the meaning uncovered by circumscription has recently prompted Shoham [88] to propose a close alternative to circumscription in which the notion of *minimal* model is replaced by an appropriate notion of *preferred* model. Shoham convincingly argues in favor of this semantic shift, and illustrates its convenience by considering a troubling problem in the domain of temporal reasoning raised by Hanks and McDermott [86].

More recently, these ideas have been further developed by Makinson [89] and Kraus *et al.* [88], who prove some interesting soundness and completeness results. Sandewal [88] has also proposed a preferential semantics for non-monotonic entailment, which he defines in terms of partial interpretations.

Nonetheless, no 'preferential semantics' attempting to capture *the intended meaning* of general default theories has yet been proposed. Defining such an account is the main goal of this paper. Our approach draws on McCarthy's [86] suggestion that default reasoning be formalized in terms of the minimization of 'abnormality.' We depart, however, from McCarthy's minimal model semantics in two ways. First, the preference ordering does not apply directly to models, but to *classes* of models, with each class embedding a commitment to certain set of assumptions. Second, the preference ordering favors classes of models which minimize *unexplained* abnormality, rather than plain abnormality. These explanations are assembled in terms of justifications which are syntactically extracted from the knowledge base. The result is an account which succeeds in eliminating the spurious models that arise in minimal model semantics, permitting a behavior in closer correspondence with intuition. In addition, the resulting framework unifies and extends ideas stemming from work in default reasoning, logic programming and abductive inference.

The paper is organized as follows. In section 2, we introduce the preference ordering. Such ordering applies to sets of models, which we call *classes*. We define the conditions under which an abnormality is regarded as *explained* in a given class, and the conditions which make a class *admissible*. In section 3, we illustrate the appeal of the proposed account by ana-

lyzing examples from the domains of reasoning about action, inheritance hierarchies, logic programming and abductive reasoning, and by comparing the results to related proposals. Finally in section 4, we summarize the main ideas, discuss some of the controversial points and point out some of the remaining problems.

2 A Preferential Semantics for Default Reasoning

2.1 Definitions

The default theories we shall consider are comprised of two components: a background context K and an evidence set E. The background context corresponds to an intensional characterization of the domain of interest in the form of a set of defeasible and undefeasible rules, while the evidence set corresponds to an extensional characterization of the particular situation of interest in the form of factual assertions [Geffner and Pearl, 87].

Among the predicate symbols occurring in the theories of interest, a distinguished set AB of predicates is used to express assumptions and abnormality conditions. In the context of default reasoning such set would contain 'abnormality' predicates [McCarthy, 86], while in the context of logic programming it would contain all the predicate symbols of interest. For a predicate $\mathtt{ab_i}$ in AB, we shall refer to atoms of the form $\mathtt{ab_i(a)}$, where a stands for a vector of ground terms, as *exceptions* or *abnormalities*, and to their negations, $\neg\mathtt{ab_i(a)}$, as *assumptions*.

The background context K of a given theory is itself structured into four components. There is a *terminological* component given by a set of strict rules (e.g. 'penguins are birds'), a *default* component given by a set of defeasible rules (e.g. 'birds fly'), a set of user-supplied *explicit exceptions* or *justifications* (e.g. 'injured birds are abnormal birds with respect to flying'), and a set of *implicit justifications* derived from the defaults in K, in a way to be described below.

Given a theory T, we are interested in characterizing the set of consequences it certifies. We shall achieve such characterization by determining the set of assumptions which can be accepted in T. For that purpose, we shall introduce the notion of a *class* C with *gap* G, given by a set of exceptions $\{\delta_1, \ldots, \delta_n\}$, as the non-empty collection of models of T which validate all the assumptions $\neg\delta$, for $\delta \notin G$.

We shall also say that a proposition holds or is validated in a class when the proposition holds in all the model members of the class. The set of all the assumptions validated by a class constitutes what we shall refer as the class *support*. Thus, in proof theoretic terms, a proposition α holds in a class C of T when the support of C comprises a set of assumptions AS such that $T, AS \vdash \alpha$.

From the complementarity of gap and support it follows that a class with a minimal gap will have a maximal support, and vice versa. Classes with minimal gaps will be referred as *minimal classes*.

As an illustration, consider for instance a theory T with a default $\mathtt{A} \wedge \neg\mathtt{ab_1} \Rightarrow \mathtt{B}$, an explicit justification $\mathtt{C} \wedge \neg\mathtt{ab_2} \Rightarrow \mathtt{ab_1}$, and a body of evidence $E = \{\mathtt{A}, \mathtt{C}\}$. For such a theory, there are no models which can make both assumptions $\neg\mathtt{ab_1}$ and $\neg\mathtt{ab_2}$ true simultaneously. Thus, there is no class of T with an empty gap. There are, however, two minimal classes C_1 and C_2, with gaps $\{\mathtt{ab_1}\}$ and $\{\mathtt{ab_2}\}$, respectively. But note that the two classes of models are not equally meritorious. Intuitively we would expect the assumption $\neg\mathtt{ab_1}$ to be defeated by the explicit justification $\mathtt{C} \wedge \neg\mathtt{ab_2} \Rightarrow \mathtt{ab_1}$, as the latter expresses a condition under which the default $\mathtt{A} \wedge \neg\mathtt{ab_1} \Rightarrow \mathtt{B}$ is not to be applicable.

Our task hereafter will be to establish a partial order among classes which will permit us to uncover the intended models of a given theory. Since each class reflects a choice of assumptions, such an ordering can be usefully regarded as a preference ordering among different assumptions sets. Thus, in the example above, we would expect the preference ordering to favor the class C_1, committed to the assumption $\neg\mathtt{ab_2}$, over the other minimal class C_2, committed to the inferior assumption $\neg\mathtt{ab_1}$. Such preference will be indeed established on the basis that the exception $\mathtt{ab_1}$ is *explained* in the class C_1, while the exception $\mathtt{ab_2}$ is not explained in the class C_2. Thus, we shall say that C_1 is *more coherent* than C_2 and is, therefore, preferred over C_2. The conditions under which an exception is explained in a class are elaborated below.

We regard default instances such as $\mathrm{A} \wedge \neg\delta \Rightarrow \mathrm{B}$ as expectations, and exceptions such as δ as expectation failures. Essentially, we assume that an exception such as δ can be explained in a class in one of two ways. Either the class validates a proposition C, in the presence of an explicit justification of the form $\mathrm{C} \Rightarrow \delta$ in K, or the class validates the propositions A and D and the assumption $\neg\delta'$, in the presence of a competing default expectation $\mathrm{D} \wedge \neg\delta' \Rightarrow \mathrm{B}'$ in K, where B′ stands for a proposition incompatible with B in K (fig. 1).

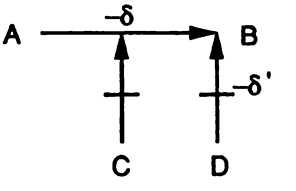


Figure 1: Explaining an exception δ

For the purposes of uniformity, however, the second case will be dealt with in a slightly different way. For each pair of defaults $\forall\mathtt{x}.\mathrm{A}(\mathtt{x}) \wedge \neg\mathtt{ab_i}(\mathtt{x}) \Rightarrow \mathrm{B}(\mathtt{x})$ and

$\forall x.D(x) \wedge \neg ab_k(x) \Rightarrow B'(x)$ with incompatible consequences in K, a formula of the form

$$\forall x.A(x) \wedge C(x) \Rightarrow ab_i(x) \vee ab_k(x),$$

referred as an *implicit justification*, will be added to K. Such formula is a deductive consequence of the rules in K, and has no effect on the models of T. Its addition, however, will permit a simple translation of the intuitions above into a formal definition.

Consider the ground instances of the rules in K. Each such ground rule might or might not involve certain assumptions in its body. For a set of assumptions AS we shall denote by K_{AS} the set of ground rules whose assumptions, if any, are among those of AS. We shall then say that an exception δ is *explained* in a class C of a context with background K and evidence E, if there is a set AS of assumptions validated by C, such that $E, K_{AS}, AS \vdash \delta$. In such case, we refer to the set AS of assumptions as an explanatory support of δ.

Note that an exception δ explained in a class C must belong to the class gap and, furthermore, it must hold in (every model of) the class. On the other hand, an exception that holds in a class, does not necessarily have an explanation in that class. Indeed, as hinted above, an exception δ can only be explained if there is a justification rule, explicit or implicit, whose consequent mentions δ. By definition, rules whose antecedents include the assumption $\neg\delta$ cannot take part of the set K_{AS} of rules needed to explain δ.

This definition of "explanation" introduces an important distinction between logically equivalent formulas, which can be illustrated by considering the rules $\neg\delta_1 \Rightarrow \delta_2$ and $\mathbf{true} \Rightarrow \delta_1 \vee \delta_2$, for two exceptions δ_1 and δ_2. The first rule permits an explanation for δ_2 in any class that validates the assumption $\neg\delta_1$. It does not permit however, an explanation of δ_1 in terms of $\neg\delta_2$; the rule $\neg\delta_1 \Rightarrow \delta_2$, with assumption $\neg\delta_1$, will belong to K_{AS} only if $\neg\delta_1$ belongs to AS. The second rule, on the other hand, does not involve any assumptions in its body and, as a result, permits explanations for δ_1 in terms of $\neg\delta_2$ and for δ_2 in terms of $\neg\delta_1$.

Recalling the example above, the reader can verify that the exception ab_1 is explained in C_1 and has an explanatory support $AS = \{\neg ab_2\}$. The exception ab_2, on the other hand, is *not* explained in the class C_2. Thus, C_1 is the only class with an empty *unexplained* gap, and it will therefore constitute the preferred class. We will make this notion more precise in the following definition.

Among two classes C and C' of a theory T, we say that C is *preferred* to C' when the unexplained gap of C is a strict subset of the unexplained gap of C'. In that case, we also say that the class C is *more coherent* than the class C'. If there is no class preferred to C, we say that C is a *preferred* class of T. Furthermore, we say that a proposition α is a consequence of T when α holds in all the preferred classes of T.

Note that it follows from these definitions that classes with smaller gaps are preferred to classes with larger gaps and, therefore, that preferred classes are always minimal. Furthermore, the preferred classes of a theory T can be determined by comparing minimal classes of T only.[2] Likewise, a class with an empty unexplained gap is always a preferred class. In these classes all exceptions are explained. We call these classes the *perfectly coherent* classes of T.

Example. Let us illustrate these definitions with the following example. We consider a theory T with a background context K comprising the following defaults (fig. 2):

1. $\forall x.\, A(x) \wedge \neg ab_1(x) \Rightarrow B(x)$
2. $\forall x.\, A(x) \wedge \neg ab_2(x) \Rightarrow C(x)$
3. $\forall x.\, B(x) \wedge \neg ab_3(x) \Rightarrow D(x)$
4. $\forall x.\, C(x) \wedge \neg ab_4(x) \Rightarrow \neg D(x)$

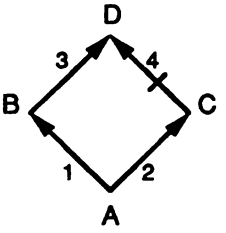


Figure 2: Simple diamond example

No undefeasible rules or explicit justifications are introduced, but the conflict between the last two defaults will result in the following implicit justification being added to K:

5. $\forall x.\, B(x) \wedge C(x) \Rightarrow ab_3(x) \vee ab_4(x).$

Note that, as we said above, such a justification is already a deductive consequence of the formulas in K. Its role is not in affecting the models of T, but in permitting the constructions of explanations for exceptions $ab_3(x)$ and $ab_4(x)$, reflecting the conflicting expectations in which they participate.

Let us now consider in T a body of evidence $E = \{A(a)\}$. The goal is to determine the preferred classes of T. There are four minimal classes C_i in this context, with gaps $\{ab_i(a)\}$ for $i = 1, 2, 3, 4$ respectively. Furthermore, the exceptions $ab_1(a)$ and $ab_2(a)$ have no explanation in the classes C_1 and C_2, as there are no justifications for these exceptions in K. On the other hand, the exceptions $ab_3(a)$ and $ab_4(a)$ are explained in C_3 and C_4 respectively, by virtue of the justification encoded by (5). As a result, we end up with two preferred and indeed perfectly coherent classes C_3 and C_4, which sanction among other conclusions, $B(a)$

[2] This will no longer be true after admissibility constraints are introduced in section 2.2.

and C(a), and which suspend judgment regarding D(a). Note that this is indeed the behavior we would expect from the diamond structure encoded by the defaults 1–4 (fig. 2). A minimal model semantics, on the other hand, would propagate the uncertainty about D(a) to the propositions B(a) and C(a) as well.

2.2 Admissible Classes

Before proceeding with more interesting cases, we must address a problem that arises from a tradeoff between exceptions and explanations induced by the proposed preference ordering. We can illustrate this tradeoff by considering a theory T, with a background comprising a default (fig. 3):

1. $\mathtt{A} \wedge \neg \mathtt{ab}_1 \Rightarrow \mathtt{B}$

and explicit justifications:

2. $\mathtt{C} \wedge \neg \mathtt{ab}_2 \Rightarrow \mathtt{ab}_1$
3. $\mathtt{B} \Rightarrow \mathtt{ab}_3$,

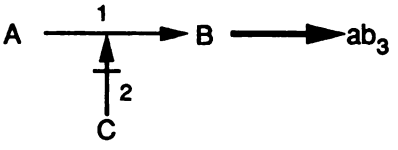


Figure 3: Spurious behavior and admissible classes

together with a body of evidence $E = \{\mathtt{A}, \mathtt{C}\}$. Such a theory gives rise to two minimal classes C_1 and C_2 with gaps $\{\mathtt{ab}_1\}$ and $\{\mathtt{ab}_2, \mathtt{ab}_3\}$, respectively. Furthermore, C_1 explains $\mathtt{ab}_1$ and C_2 explains $\mathtt{ab}_3$. The exception $\mathtt{ab}_2$, on the other hand, has no explanation in C_2. It follows then, that C_1, unlike C_2, has an empty unexplained gap and, therefore, that C_1 is the preferred class. This in turn can be interpreted as indicating, in agreement with intuition, that the proposition C defeats the default $\mathtt{A} \wedge \neg \mathtt{ab}_1 \Rightarrow \mathtt{B}$.

Consider now the case in which the exception $\mathtt{ab}_3$ is incorporated into the current evidence pool, so that the total evidence becomes $E = \{\mathtt{A}, \mathtt{C}, \mathtt{ab}_3\}$. In such a context, again, two minimal classes C_1' and C_2' arise; the former with a gap $\{\mathtt{ab}_1, \mathtt{ab}_3\}$, and the latter with a gap $\{\mathtt{ab}_2, \mathtt{ab}_3\}$. As before, $\mathtt{ab}_1$ is explained in C_1' and $\mathtt{ab}_3$ is explained in C_2'. Nonetheless, in the current context, neither class turns out to be preferred over the other. As a result, unexpectedly, the introduction of the exception $\mathtt{ab}_3$ has the effect of reinstating the default encoded by 1, which is no longer defeated.

This spurious effect can be explained in terms of the abductive bias embedded in the preference ordering, by which classes capable of explaining their exceptions are rewarded. In this case, the reinstatement of the assumption $\neg \mathtt{ab}_1$ permits the construction of an explanation for the exception $\mathtt{ab}_3$, but comes at the price of introducing the *unexplained* exception $\mathtt{ab}_2$. This tradeoff can be shown indeed to underlie this and other forms of abnormal behavior arising from the proposed preference ordering. In what follows a restriction on classes will be defined which will rule out such type of situations. Classes to be considered will have to be *admissible* in the sense defined below.

First, let us say that a class C of T with gap G *supersedes* a class C' of T with gap G', when the set $G - G'$ is not empty and only contains exceptions explained in C, while the set $G' - G$ contains exceptions unexplained in C'.

Thus, the gap of a class C' superseded by a class C can be constructed by eliminating some explained exceptions from C's gap, and by adding new exceptions, not all of them explained in C'. In terms of the example above, it can be verified that the class C_1' with gap $\{\mathtt{ab}_1, \mathtt{ab}_3\}$ supersedes the class C_2' with gap $\{\mathtt{ab}_2, \mathtt{ab}_3\}$. The latter gap can indeed be obtained from the former by removing the explained exception $\mathtt{ab}_1$ and by adding the unexplained exception $\mathtt{ab}_2$.

Finally, a class is *admissible* when it is not superseded by any other class. Hereafter, preferred classes will be selected by considering admissible classes only.

3 Applications

In the previous section we have laid out a semantic framework for the characterization of default theories. Our goal in this section is to illustrate how such a framework applies to a variety of domains ranging from problems in temporal reasoning, to problems in inheritance hierarchies, logic programming and abductive reasoning. Special emphasis will be placed on the type of behavior legitimized by the proposed account. Recall that our main goal is to arrive at an interpretation of the theories of interest which better approximates the intended interpretation.

3.1 Reasoning about Action

Our appeal to coherence considerations in pruning the set models of a given theory makes the proposed framework closely related to the proposals of Lifschitz [87], Haugh [87] and Morgenstern and Stein [88] for formalizing reasoning about action. In these proposals, clippings (persistence exceptions) can only originate from acting causes. Lifschitz and Haugh then minimize then over these causes, subject to explaining the clippings. Morgenstern and Stein take a slightly different view and select those models in which the actions are causally 'motivated' by the available evidence.

In our proposal, we do not require a cause behind every clipping, but 'reward' those classes of models in which this is the case and, therefore, those classes in which clippings are explained. We thus avoid some undesirable features of these approaches (Lifschitz's and Haugh's ontologies and Morgenstern's and Stein's lim-

itation to accommodate defeasible causal rules), while obtaining an additional degree of flexibility.

Consider the following version of the 'Yale Shooting Problem' raised by Hanks and McDermott [86]. We have a theory T with a background context K given by the following expressions:[3]

1. $\forall t.\, \mathtt{LD}(t) \Rightarrow \mathtt{LDD}(t+1)$
2. $\forall t.\, \mathtt{LDD}(t) \wedge \neg \mathtt{ab}_1(t) \Rightarrow \mathtt{LDD}(t+1)$
3. $\forall t.\, \mathtt{ALV}(t) \wedge \neg \mathtt{ab}_2(t) \Rightarrow \mathtt{ALV}(t+1)$
4. $\forall t.\, \mathtt{SHT}(t) \wedge \mathtt{LDD}(t) \wedge \neg \mathtt{ab}_3(t) \Rightarrow \neg \mathtt{ALV}(t+1)$
5. $\forall t.\, \mathtt{SHT}(t) \wedge \mathtt{LDD}(t) \Rightarrow \mathtt{ab}_2(t)$

Thus, we have that a loading event makes the gun loaded, that loaded guns remain loaded and that alive 'animals' remain alive unless shot with a loaded gun. Furthermore, due to the conflict between the persistence of 'alive' (3) and the shooting rule (4), an implicit justification of the following form is added to K:

6. $\forall t.\, \mathtt{SHT}(t) \wedge \mathtt{LDD}(t) \wedge \mathtt{ALV}(t) \Rightarrow \mathtt{ab}_3(t) \vee \mathtt{ab}_2(t)$

The evidence indicates that a turkey called Fred was alive at time $t = 1$, that a loading event took place at time $t = 2$, and that a shooting event directed at Fred took place at $t = 3$. Intuitively, it appears that Fred should no longer be alive as a result of the shooting. However, as Hanks and McDermott noted, several minimal classes pop up, in some of which Fred survives the shooting. In our formulation, these are classes in which the gun is mysteriously unloaded or in which the shooting, for some reason, misses its target. The collection of minimal classes of T thus corresponds to classes with a single exception among $\mathtt{ab}_1(2)$ ('mysterious unloading'), $\mathtt{ab}_2(1)$, $\mathtt{ab}_2(2)$ ('mysterious death'), $\mathtt{ab}_2(3)$ ('death by shooting'), and $\mathtt{ab}_3(3)$ ('target missed'). It is not difficult to show that the class corresponding to $\mathtt{ab}_2(3)$ is the only perfectly coherent class, and is thus the single preferred class of T. This is due to the fact that the persistence exception $\mathtt{ab}_2(3)$ can be explained in terms of the explicit justification encoded by (5). None of the other exceptions, on the other hand, can be given an explanation.

An undesirable feature of the above formulation which is shared by Hanks' and McDermott's, is the need to explicate by means of an explicit justification (5), that the shooting rule is supposed to prevail over the persistence of 'alive' (3). This is done by declaring the latter persistence to be abnormal in the context of a shooting. In a more realistic setting though, where a number of events leading to different type of changes are to coexist, the number of intended 'clippings' which must be enumerated could be overwhelming. Moreover, these explicit exceptions, as we show below, make theories less modular.

Consider for instance the possibility that Fred was wearing a metal vest at the time of the shooting. We could describe the effect of wearing a metal vest by asserting that the shooting rule is not applicable to somebody wearing a metal vest:

7. $\forall t.\, \mathtt{VEST}(t) \Rightarrow \mathtt{ab}_3(t)$

Such rule, however, would fail to achieve its intended effect. While the death of Fred would no longer follow, the justification encoded by (5) would still prevent proving Fred alive after the shooting. This suggests that a more flexible means of specifying the intended priority of rules about change is needed.

Our proposal is a simple one, consisting of two parts. First, we allow the user to lexically distinguish the assumptions associated with rules about change from the assumptions associated with rules about persistence.[4] We do so by replacing the generic type of normality assumption $\neg \mathtt{ab}_i(\cdot)$ with two different types of assumptions: $\neg \mathtt{cp}(\cdot)$, read 'not clipped', which is used for assumptions about persistence; and $\neg \mathtt{pv}(\cdot)$, read 'not prevented', which is used for assumptions about the result of actions. The formulation above would then be translated into the more concise description:

1'. $\forall t.\, \mathtt{LD}(t) \Rightarrow \mathtt{LDD}(t+1)$
2'. $\forall t.\, \mathtt{LDD}(t) \wedge \neg \mathtt{cp}_1(t) \Rightarrow \mathtt{LDD}(t+1)$
3'. $\forall t.\, \mathtt{ALV}(t) \wedge \neg \mathtt{cp}_2(t) \Rightarrow \mathtt{ALV}(t+1)$
4'. $\forall t.\, \mathtt{SHT}(t) \wedge \mathtt{LDD}(t) \wedge \neg \mathtt{pv}_3(t) \Rightarrow \neg \mathtt{ALV}(t+1)$

where the priority of the shooting rule over the aliveness persistence rule is not explicated.

As usual, the conflict between the last pair of default rules results in the addition of a corresponding implicit justification to K. Recall that these implicit justifications permit us to explain the failure of certain expectation in terms of the success of an alternative, incompatible expectation. Now, however, we have *two different types of expectations:* we have expectations of change on the one hand, and expectations of persistence on the other. The two expectations, however, are not intended to be treated symmetrically. While it is assumed that a successful change explains a corresponding clipping, it is also assumed that a failed

[3] Note that, unlike Hanks and McDermott, we have expressed the shooting rule (5) as a default rule. Independently of whether this is a more appropriate encoding, such a choice is motivated by the assumption embedded in the definition of 'explanations', by which expectations are assumed to be encoded by defaults. A slight extension would be needed to accommodate, for instance, undefeasible causal rules. We shall not pursue that extension in this paper. Let us just point out, however, that the shooting rule could be made undefeasible by simply declaring the proposition $\neg \exists t.\, \mathtt{ab}_3(t)$ as part of the evidence.

[4] If we were using a reified temporal notation in the style of [Shoham, 88], a single persistence rule would suffice. Nonetheless, in order to simplify the description of the example, we have found a non-reified notation more convenient and, therefore, a collection of persistence rules is needed.

action is not to be explained in terms of the persistence it fails to clip. We incorporate this asymmetry into our account by defining the implicit justifications associated with the conflict of a rule about change and a rule about persistence in a different manner. Thus, from the conflict between defaults (3′) and (4′) above, rather than eliciting the implicit justification:

$$5''.\ \forall t.\ \mathtt{SHT}(t) \wedge \mathtt{LDD}(t) \wedge \mathtt{ALV}(t) \Rightarrow \mathtt{pv}_3(t) \vee \mathtt{cp}_2(t)$$

we assert the logically equivalent, but asymmetric justification:

$$5'.\ \forall t.\ \mathtt{SHT}(t) \wedge \mathtt{LDD}(t) \wedge \mathtt{ALV}(t) \wedge \neg \mathtt{pv}_3(t) \Rightarrow \mathtt{cp}_2(t)$$

Thus, we allow the clipping of 'alive', $\mathtt{cp}_2(\cdot)$, to be explained in terms of a 'successful' shooting, but preclude the 'successful' persistence of 'alive' from explaining an 'unsuccessful' shooting, $\mathtt{pv}_3(\cdot)$. The reader can verify that from a theory with the background context defined by the formulas 1′–5′, and given the same evidence as above, the same conclusion about Fred follows. The difference now is that certain preferences are handled implicitly, and that the resulting formulation is more flexible. The metal vest variation, for instance, would work without modification in this setting.

3.2 Inheritance Hierarchies

Another area in which minimal model semantics falls short of delivering the intended models of a set of defaults is in the context of inheritance hierarchies. Inheritance hierarchies are convenient devices for organizing knowledge about prototypical classes of individuals. Rather than explicitly stating the attributes of each possible individual, individuals are assumed to implicitly inherit a certain set of attributes by virtue of the place they occupy in the hierarchy. The key problem to address in these structures arises when an object belongs to classes with incompatible attributes. The typical example goes like this: Tweety is a penguin and, therefore, a bird. Typically birds fly and typically penguins do not fly. What should be concluded about the flying abilities of Tweety?

It is commonly accepted that there is an *implicit* preference among the defaults represented in these networks. Such preference appears to establish a priority for defaults rooted in more specific information [Touretzky, 86]. In terms of the example above, such preference would favor for instance the belief that Tweety is likely not to fly, on the basis that penguins are a subclass of birds. For more complex cases, the preferences are not always so clear, though significant progress has been made in recent years, both in the context of inheritance hierarchies [Horty *et al.*,87] and in more general settings (e.g. [Loui, 87; Delgrande, 87; Geffner and Pearl, 87]).

An important insight into the nature of the intended preferences among conflicting defaults that has emerged from these proposals is that a default 'if A then B' constitutes a license to infer B when A represents *all* the available evidence. In other words, a default antecedent provides a safe context on which the truth of the default consequence can be asserted. We refer to this aspect of defaults as the *context sensitivity* property of defaults.

In the context of the framework we have been developing, accounting for the context sensitivity of a default 'if A then B' would amount to making B true in all the preferred classes of A.[5] With that goal in mind, we shall impose a further restriction on the classes to be considered when dealing with theories, such as inheritance hierarchies, where there is an implicit preference to be uncovered among defaults.

Let $A \wedge \neg\delta \Rightarrow B$ be the ground instance of a default in K such that $K, A \not\vdash \delta$. We say that a set of assumptions $AS = \{\neg\delta_1, \ldots, \neg\delta_n\}$ is in *conflict* with the assumption $\neg\delta$, if the assumptions in AS compete with $\neg\delta$ upon learning A, i.e. if $K, A \vdash \delta \vee \delta_1 \vee \cdots \vee \delta_n$. Since $\neg\delta$ is the intended assumption in such context, it is reasonable to assume that the user intends to reject some of the assumptions in AS. We say then, that the set of assumptions AS is *dominated* by A. A *context-admissible* class C is then defined simply as an admissible class which does not validate any assumption set dominated by propositions that hold in C.[6]

For inheritance theories, only context-admissible classes will be considered. Note that in order to test context-admissibility, it is sufficient to examine *minimal* dominated assumption sets only.

Example. This example illustrates the type of specificity preferences entailed by the context-admissibility restriction. Let T be a theory with a background context K given by the following defaults (fig. 4):

$$\forall x.\ \mathtt{A}(x) \wedge \neg \mathtt{ab}_1(x) \Rightarrow \mathtt{B}(x)$$
$$\forall x.\ \mathtt{B}(x) \wedge \neg \mathtt{ab}_2(x) \Rightarrow \mathtt{C}(x)$$
$$\forall x.\ \mathtt{B}(x) \wedge \neg \mathtt{ab}_3(x) \Rightarrow \mathtt{D}(x)$$
$$\forall x.\ \mathtt{C}(x) \wedge \neg \mathtt{ab}_4(x) \Rightarrow \neg\mathtt{D}(x)$$
$$\forall x.\ \mathtt{F}(x) \wedge \neg \mathtt{ab}_5(x) \Rightarrow \mathtt{C}(x)$$

Due to the conflict between the defaults associated with the assumptions $\mathtt{ab}_3(x)$ and $\mathtt{ab}_4(x)$ (fig. 4), the following implicit justification will also be part of K:

$$\forall x.\ \mathtt{B}(x) \wedge \mathtt{C}(x) \Rightarrow \mathtt{ab}_3(x) \vee \mathtt{ab}_4(x)$$

We consider a body of evidence $E = \{\mathtt{A}(a), \mathtt{F}(a)\}$.

[5] Kraus *et al.* [88] interpret defaults in a similar manner. Selman and Kautz [88], on the other hand, account for specificity preferences by intepreting defaults as imposing an ordering over pairs of models.

[6] Let us point out that in a pathological net with default instances $A \wedge \neg\delta \Rightarrow B$ and $A \wedge \neg\delta' \Rightarrow \neg B$, a context-admissible class would be forced to reject both assumptions $\neg\delta$ and $\neg\delta'$ upon learning A. With good reason, such networks are inconsistent in the frameworks of Horty *et al.* [87], Delgrande [87] and Geffner and Pearl [87].

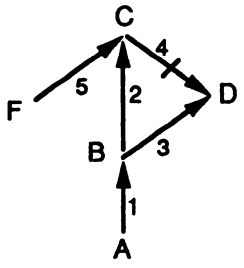


Figure 4: A simple inheritance hierarchy

Intuitively, we would expect the preferred classes of T to sanction the propositions $\mathtt{B(a)}$, $\mathtt{C(a)}$ and $\mathtt{D(a)}$. There are however four minimal classes of T, among which are two perfectly coherent classes C_3 and C_4, with explained gaps $\{\mathtt{ab_3(a)}\}$ and $\{\mathtt{ab_4(a)}\}$ respectively. C_4 represents the intended class of models, while C_3, which sanctions $\neg\mathtt{D(a)}$, fails to embed the right specificity preferences. We show below that C_3 is not a context-admissible class.

Consider the default instance $\mathtt{B(a)} \wedge \neg\mathtt{ab_3(a)} \Rightarrow \mathtt{D(a)}$. It follows from the body of defaults in K that the assumption set $AS = \{\neg\mathtt{ab_2(a)}, \neg\mathtt{ab_4(a)}\}$ is in conflict with the assumption $\neg\mathtt{ab_3(a)}$, and therefore, that the set AS is dominated by the proposition $\mathtt{B(a)}$. The class C_3, however, validates both $\mathtt{B(a)}$ and AS, and thus is not context-admissible. This leaves C_4, the intended class, as the single preferred class of T.

Let us remark that even if we add the following explicit justification to T:

$$\forall \mathtt{x}.\, \mathtt{B(x)} \Rightarrow \mathtt{ab_4(x)}$$

a simple minimization of abnormality would still not yield the expected conclusions in this case. Among the minimal models of T there would still be models validating both the abnormality $\mathtt{ab_1(a)}$ and the proposition $\neg\mathtt{B(a)}$.

3.3 Closed World Reasoning

The reader may have noticed that the preference ordering introduced above does not involve a minimization of the *extension* of the (abnormality) predicates in AB, but rather, a minimization of the set of *truths*. In other words, the gap of a class is not defined in terms of the exceptional individuals in the domain, but in terms of a set of ground exceptions. As a result, and in contrast to minimal model semantics, a default like 'birds fly' allow us to conclude that 'Tweety flies' upon learning that 'Tweety is a bird', without ever being committed to the conclusion that 'all birds fly'. A model in which only certain unnamed birds do not fly is as preferred as a model in which all birds fly, and both models would be indeed part of the same preferred class.

The same choice also permits jumping to conclusions without the need of unique name axioms [Reiter, 80b]. If Tim is a penguin, we do not need to prove that Tweety is different from Tim in order to jump to the conclusion that Tweety flies. This again contrasts with the form the same inference would be certified by a minimal model semantics, in which such inequality would be required.

Both of these features suggest that, in terms of jumping to conclusions, the proposed framework bears a closer similarity to Reiter's default logic than to McCarthy's circumscription. Furthermore, as we shall see, the framework inherits a difficulty of Reiter's logic in handling conclusions regarding unbounded sets of individuals, pointed out by McCarthy in [McCarthy, 80].

Let us say, for instance, that we want to capture the type of default behavior found in some relational database. We might have in the database a collection of tuples of the form $\mathtt{P(a)}, \mathtt{P(b)}$ and $\mathtt{Q(a)}, \mathtt{Q(b)}, \mathtt{Q(c)}$. From such database, conclusions like as that "a and b are the only P's," or that "all the P's are Q's" would follow. These are conclusions that non-monotonically depend on the state of the database, and that can potentially be defeated by the acquisition of new tuples (e.g. $\mathtt{P(d)}$).

A minimal model semantics would have no difficulty in accounting for such behavior. A simple minimization of the extensions of P and Q, together with the appropriate unique-name axioms will do. In our framework however, the straightforward approach of declaring the predicates P and Q as 'abnormality' predicates, members of AB, would not quite work. From such a declaration, we could derive conclusions such as $\neg\mathtt{P}(x)$, for any x different than a and b, but not *universals* such as $\forall \mathtt{x}.\, \mathtt{P(x)} \Leftrightarrow \mathtt{x} = \mathtt{a} \vee \mathtt{x} = \mathtt{b}$, which involve a commitment regarding unnamed individuals in the relevant models. In the remainder of this subsection we show that it is possible to capture this type of *closed world reasoning* in the present setting. The key, as hinted in [McDermott, 82], consists of incorporating *sets* into the universe of discourse. We shall not elaborate here on the details of how such an extension can be defined; suffice it to say that any weak set theory will do.[7]

In order to illustrate how the behavior of the database described can be captured in terms of defaults involving reference to sets of individuals, we shall introduce the following two abbreviations:

$$\begin{aligned} \mathtt{P[S]} &: \forall \mathtt{x}.\, \mathtt{x} \in \mathtt{S} \Rightarrow \mathtt{P(x)} \\ \hat{\mathtt{P}}\mathtt{[S]} &: \forall \mathtt{x}.\, \mathtt{x} \in \mathtt{S} \Leftrightarrow \mathtt{P(x)} \end{aligned}$$

where S stands for an arbitrary set of individuals. Thus, $\hat{\mathtt{P}}$ represents the definition, or as we shall we say, the 'closed' version of P. Having these abbreviations available, we can capture the database behavior by a theory with background:

[7] The interested reader might want to consult [Perlis, 88] for a relevant discussion.

$$\forall S. \quad P[S] \wedge \neg ab_1(S) \Rightarrow \hat{P}[S]$$
$$\forall S, S'. \quad P[S] \wedge S \supset S' \Rightarrow ab_1(S')$$

That is, if the members of a set S' are all instances of P, then it is assumed that S' contains *all* the instances of P *unless* there is a larger set S whose members are also instances of P. We shall also need a unique name hypothesis in order to distinguish different sets:

$$\forall x, y. \neg ab_2(x, y) \Rightarrow x \neq y$$

Notice that this default introduces a set of unexplained exceptions of the form $ab_2(x, x)$ in every class. We refer below to these exceptions as the common exceptions.

We can now analyze different states of the database as well as the conclusions which are sanctioned in each case.

Case 1. $E = \emptyset$. Without any tuples in the database, the preferred class includes no exception in addition to the common exceptions mentioned above. Thus, $\neg ab_1(\emptyset)$ forms part of the support of the preferred class and, therefore, $\hat{P}[\emptyset]$ and $\neg\exists x. P(x)$ follow.

Case 2. $E = \{P(a)\}$. $ab_1(\emptyset)$ becomes an explained exception, part of the gap of the preferred class. Still, the assumption $\neg ab_1(\{a\})$ holds, and the conclusion $\hat{P}[a] : \forall x. P(x) \Leftrightarrow x = a$ follows.

Case 3. $E = \{P(a), P(b)\}$. Now the gap of the preferred class is enhanced by two new explained exceptions $ab_1(\{a\})$ and $ab_1(\{b\})$. The assumption $\neg ab_1(\{a, b\})$ still holds, so that $\hat{P}[a, b] : \forall x. P(x) \Leftrightarrow x = a \vee x = b$ follows.

Case 4. $E = \{P(a) \vee P(b)\}$. In this context, the preferred class includes only the explained exception $ab_1(\emptyset)$ in addition to the common exceptions. Its models can be divided into two sets: those in which $P(a) \wedge \neg P(b)$ holds, and those in which $P(b) \wedge \neg P(a)$ holds. In the first class of models, $\hat{P}[a]$ holds as well, and in the second set of models, $\hat{P}[b]$ does. As a result, the disjunction $\hat{P}[a] \vee \hat{P}[b]$, which abbreviates the expression $[\forall x. P(x) \Leftrightarrow x = a] \vee [\forall x. P(x) \Leftrightarrow x = b]$, holds in the preferred class.

These results illustrate that it is possible to capture in the present framework the form of closed world reasoning found in databases. Circumscription will sanction the same conclusions in each of these cases [Lifschitz, 85]. In other cases, however, the results might differ. One such case, for instance, would correspond to $E = \{\exists x.P(x)\}$. Given such a context, circumscription would conclude that there is a single instance of P, i.e. $\exists x.\forall y.P(y) \Leftrightarrow x = y$. However, such a conclusion does not follow from the account presented.

3.4 Logic Programming

The semantic framework proposed can also be applied to logic programs with negation (see [Shepherson, 88] for a review). For logic programs, the set AB of predicates whose truth sets are expected to be minimal is identical to the set of all predicates of interest. A logic program is thus a collection of what we have been calling explicit justifications. In what follows, for ease of comparison with other proposals, we consider only *Herbrand* models.

The first result we show relates the proposed semantics to the stable semantics for logic programs proposed by Gelfond and Lifschitz [88] and, independently, by Kit Fine [88]. Elsewhere [Van Gelder *et al.*, 88], it has been proved that the stable model of stratified logic programs is unique, and that it coincides with the canonical model of Apt *et al.* [88] and the perfect model of Przymusinski [88].

Following Gelfond and Lifschitz we assume that each rule in the program P of interest has been replaced by all its ground instances. A Herbrand model M of P is then defined as stable if and only if M is the minimal model of the program P_M^+. P_M^+ is the positive program obtained by removing from P all the rules whose bodies contain assumptions $\neg\delta$, with $\delta \in M$, and by deleting the assumptions (i.e. negative literals) from the remaining rules.

If M is a model of a program P, we will denote by C_M the class of models of P with a gap equal to M. Thus M, as well as models of P smaller than M, will belong to C_M. The following theorem then holds:[8]

Theorem 1. *M is stable if and only if the class C_M is perfectly coherent.*

In words, the theorem says that M is stable if each atom of M has an explanation in terms of the assumptions validated by M. Note that since a stable model is always minimal, the class C_M will contain a single model, namely M.

Still, there are programs which have no stable models. These programs might nonetheless have a well defined set of preferred classes. One typical example is the program P, composed of the single clause $p \Leftarrow \neg p$. The preferred class of P has the single *unexplained* atom p in its gap.

The correspondence between stable models and perfectly coherent classes suggests that the criterion of stability which is used in defining the stable semantics of logic programs embeds an abductive bias by which models capable of explaining their atoms are rewarded. This feature becomes apparent when we consider the following two programs:[9]

$$P_1: \begin{array}{l} q \Leftarrow \neg r \\ r \Leftarrow \neg q \\ p \Leftarrow \neg p \\ p \Leftarrow \neg r \end{array} \qquad P_2: \begin{array}{l} q \Leftarrow \neg r \\ p \Leftarrow \neg q \\ p \Leftarrow \neg p \end{array}$$

In both programs, the clause $p \Leftarrow \neg p$ introduces, but does not explain, the atom p. This leads the stable semantics to produce results in both cases which differ

[8] Proofs are omitted due to lack of space. They can be obtained by writing to the author.

[9] P_1 is taken from [Van Gelder *et al.*, 88].

from those which would be obtained if this clause were replaced by the simpler clause $p \Leftarrow$. The problem is that our preference ordering, as well as the stable semantics for logic programs, rewards those classes in which p *gets* an explanation. Thus in P_1, both semantics favors the model $M_1 = \{q, p\}$ over the apparently equally meritorious model $M_1' = \{r, p\}$, while in P_2, the apparently superior model $M_2 = \{q, p\}$ fails to receive a better ranking than the model $M_2' = \{r, p\}$.

These examples appear to suggest that a more intuitive preference criterion for selecting the intended models of general logic programs should have this abductive bias removed. We discussed the effects of such a bias when the admissibility restriction was introduced in section 2.2. Recall that an admissible class is a class not superseded by any other class. Likewise, a class C' with gap G' is superseded by a class C with gap G when the set $G - G'$ is non-empty and only contains exceptions explained in C, while the set $G' - G$ contains exceptions unexplained in C',

In the examples above, the *minimal admissible classes* turn out to be in precise correspondence with the more intuitive models M_1, M_1' and M_2. The class with gap M_2', on the other hand, is superseded by the class with gap M_2, and is therefore not admissible.

Interestingly enough, the perfectly coherent class of a stratified program is also its unique minimal admissible class. That is, there is a correspondence between the minimal admissible class of a stratified program P and the canonical model of P, as defined by Apt *et al.*, Przymusinski and others. This correspondence is summarized in the following theorem:

> **Theorem 2.** For a stratified program P, there is a unique minimal admissible class, whose gap is the canonical model of P.

Thus we have two alternative semantics for general logic programs: one based on the preference ordering formerly introduced, the other which simply selects the minimal admissible classes. Both semantics coincide for the family of stratified programs, but diverge outside that family. The examples above suggest that a semantics based on minimal admissible classes is free from the abductive bias exhibited by the stable semantics and the preferential semantics here proposed and, therefore, that it might constitute a more appropriate basis for identifying the intended model(s) of general logic programs.

As a final illustration, we will consider a program P in which none of the minimal classes is admissible and in which, therefore, the preferred class is non-minimal. P is given by the following rules:

$$p \Leftarrow \neg q$$
$$q \Leftarrow \neg r$$
$$r \Leftarrow \neg p$$

The minimal Herbrand models of P are $M_1 = \{p, r\}$, $M_2 = \{q, p\}$ and $M_3 = \{q, r\}$, while the minimal classes are C_{M_1}, C_{M_2} and C_{M_3}. It can be shown that none of these classes is admissible: C_{M_2} supersedes C_{M_1}, C_{M_3} supersedes C_{M_2}, and C_{M_1} supersedes C_{M_3}. Thus, the minimal admissible class of P which is also the admissible class favored by preference ordering turns out to be the non-minimal class C_M, with gap $M = \{p, q, r\}$. C_M stands in this case for a *collection of models;* indeed, it represents the set of subsets of M which are models of P, i.e. $C_M = \{M, M_1, M_2, M_3\}$.[10] It follows then, that none of p, q, r, or any of their negations are sanctioned as consequences of P.

3.5 Abductive Reasoning

Work in non-monotonic reasoning has been inspired by the goal of providing a formal account of some of the pervasive patterns of inference found in commonsense reasoning. Most of this work to date has been focused on the characterization of what has been called default inference, a form of reasoning akin to deductive inference, in which certain assumptions are adopted in the absence of contrary evidence. Nonetheless, other forms of non-monotonic inference, qualitatively different from default reasoning, also appear to play an important role in commonsense inference. One such form, analyzed in some detail in [Harman, 86], is what has been variously referred to as "inference to the best explanation," "abductive reasoning" or "conjectural reasoning." This is a form of inference which attempts to make sense of the evidence by increasing the coherence of a given set of beliefs. The characterization of these patterns of inference involves both the determination of sources of incoherence in a given belief state and the identification of hypotheses capable of explaining such incoherence away. In this subsection, we shall attempt to show that the framework we have so far developed lends itself to a characterization of this sort.

We assume that the unexplained gaps associated with the preferred classes of a given context provide a useful measure of the coherence of such context; indeed, they point out 'what needs to be explained.' For instance, in a inheritance hierarchy about animals, a context which mentions a bird Tweety that does not fly would be slightly incoherent. In such an incoherent state, it might make sense to jump to conclusions which could explain away the source of incoherence. We could hypothesize for instance, that Tweety is sick, or that he is penguin and so on. We shall refer to those propositions as *conjectures*. More precisely, a ground

[10] Recall from section 2, that for a theory T, a class C with gap G stands for the non-empty collection of models of T which validate all the assumptions $\neg\delta$, for $\delta \notin G$. C_M thus represents the collection of models of P which validate all the literals $\neg\alpha$, for $\alpha \notin M$, i.e. all the Herbrand models of P included in M.

atomic proposition γ would be regarded as a conjecture in a context T if its adoption yields a new context $T \cup \{\gamma\}$ *more coherent* than the original context T.

In section 2.1, we defined the conditions that make one class more coherent than another. Now we must define a similar order between *contexts*. For this purpose, we will associate with every context T a coherence descriptor $H[T]$, given by the vector of unexplained gaps in its preferred classes. A context T with a coherence descriptor $[G_1, \ldots, G_n]$ would then be said to be *as coherent* as a context T' with a coherence descriptor $[G'_1, \ldots, G'_m]$, if each G_i, $1 \leq i \leq n$, is included in G'_j, for some j, $1 \leq j \leq m$. Furthermore, if T is as coherent as T' but T' is not as coherent as T, we say that T is *more coherent* than T'.

Recalling, the example above, we obtain then that the proposition 'Tweety is a penguin' would qualify as a conjecture in the above context, since its adoption would lead to a context whose preferred classes are perfectly coherent. Nonetheless, a proposition such as 'Tweety is a brown arctic penguin' would also qualify as a conjecture. In order to rule out unnecessarily specific conjectures, we must restrict the space of admissible conjectures. Let us say that a conjecture γ' is less specific than a conjecture γ in a context T if γ' follows from $T \cup \{\gamma\}$, but γ does not follow from $T \cup \{\gamma'\}$. Then, we say that a conjecture γ is *admissible* if there are no less specific conjectures leading to contexts as coherent as those resulting from the adoption of γ. Thus, while 'Tweety is a penguin' and 'Tweety is sick' would represent admissible conjectures in the above context, the proposition 'Tweety is a brown arctic penguin' would not.

Note that there is an important distinction between the set of default conclusions that follow from a given theory and the set of admissible conjectures legitimized by it. The set of admissible conjectures, unlike the set of default consequences, is not deductively closed. Indeed, while it is reasonable to conjecture that Tweety does not fly because it is sick, or because it is a penguin, it is not so reasonable to conjecture that Tweety does not fly because it is a sick penguin. Conjectures, unlike defaults, represent *alternative* belief changes.

While this account of conjectural reasoning does not limit the space of admissible conjectures a priori,[11] it may nonetheless be useful to provide the user with the facility of expressing conditions under which certain conjectures would be preferred over others. For instance, following Pearl [87], we might want to express things like 'if you are not aware of an explanation for the grass being wet, then conjecture that it rained'. Thus, if the grass is observed to be wet, even in the presence of other admissible conjectures, an explanatory conjecture stating that it rained would be adopted. However, if an alternative explanation is learned, say that the sprinkler was on, the reason for having postulated 'rain' would vanish, and so the 'rain' conjecture. Pearl refers to these defaults as *evidential* defaults, and convincingly argues that they require a treatment different from the one given to normal (causal) defaults. In our framework, evidential defaults turn out to be essentially context guided conjectures. The details on how they can be accommodated as part of the language are elaborated in [Geffner, 89].

[11] See [Poole, 87] for a different view.

4 Discussion

We have presented a framework for characterizing defeasible inference based on a preference ordering among classes of models. The ordering favors classes with a minimal unexplained set of exceptions. We have shown how such a framework permits to unify ideas stemming from work in default reasoning, logic programming and abductive inference. We have also illustrated how the proposed account eliminates the spurious models that arise in minimal model semantics, permitting a behavior in closer correspondence with intuition.

The account presented here is unorthodox in several ways. First, a preferential ordering is described which does not apply directly to models, but to classes of models. The motivation for such a choice originates from viewing default reasoning in the 'abnormality' setting as a labeling problem, in which the set of legitimate assumptions in a given context needs to be identified. Each class of models thus represents a choice of assumptions, and these choices are evaluated according to the preference ordering.[12]

The distinction between explained and unexplained abnormality plays a central role in such ordering. We have argued that the value of a class is not in inverse proportion to its abnormalities, but rather to its *unexplained* abnormalities. No penalty, for example, we have maintained should be associated with a class in which a bird does not fly, if the bird is, say, a penguin. In that situation, being an 'abnormal bird with respect to flying' is the normal, expected condition. Abnormalities are unlikely in certain contexts but likely in others, and a reasonable preference ordering should be able to make this distinction.

Our reliance on justifications which are syntactically extracted from the database and used to construct ex-

[12] This view also suggests an alternative, stronger definition of default entailment which we have not pursued in this paper. Rather than defining α to be a default consequence of T when α holds in all the preferred classes of T (section 2.1), we could require the existence of a set of assumptions AS validated in all the preferred classes of T, such that $T, AS \vdash \alpha$. This stronger definition appears to bear some resemblance with those semantic accounts based on partial models (e.g. [Van Gelder *et al.*, 88]), which we have not yet investigated.

planations is potentially more controversial. We have assumed that abnormalities represent expectation failures, and as such, could be either explained by explicit exceptions which assert that a default is not applicable in a given circumstance, or by competing expectations. This choice, however, is not unique and, quite possibly is not the best. It is, however, relatively simple and intuitive, and as we have illustrated, it can 'reasonably' account for 'reasonable' examples. Further refinements may still be necessary.

The framework for defeasible inference proposed here shares several features with the system **L** proposed in [Geffner, 88]. Both systems represent defaults in the same way and they appeal to the same distinction between background and evidence. Both regard the antecedent of a default as providing a safe context in which the truth of the consequent can be asserted, and both attempt to capture the distinction between defaults and their contrapositives in a similar way.

There are, nonetheless, significant differences between the two frameworks. First, **L** has the form of a natural deduction system whose rules originate from a probabilistic interpretation of defaults.[13] This set of rules is supplemented by an additional, more ad-hoc rule, which attempts to supply the probabilistic rules with appropriate assumptions about conditional independence. Such an "irrelevance rule," as it is called, permits us to infer for instance conclusions like 'a red bird flies,' given that 'birds fly'. These conclusions would otherwise escape the probabilistic machinery.

The irrelevance rule in **L**, and an analogous construction in the conditional logic of Delgrande [87], plays a central role in endowing these systems with a reasonable inferential power. There has been, however, a difficulty in justifying and making precise the form this rule should take. This difficulty has been a primary motivation behind the work reported in this paper. The framework we have elaborated here provides a rationale for identifying the set of assumptions to adopt in a given context.

Nonetheless, the proposed semantics does not validate the probabilistic rules of **L**; indeed, unlike **L**, the semantics is not *cumulative*.[14] In other words, even if H defeasible follows from T, the contexts T and $T \cup \{H\}$ are not guaranteed to yield the same conclusions.[15] In this regard, two important questions remain to be answered. The first has to do with whether 'cumulativity' is a reasonable property to have in a defeasible logic, and if so, whether it is possible to embed a cumulative logic within a semantics capable of drawing sensible assumptions about conditional independence. A discussion of some of the issues involved in these questions can be found [Geffner, 89].

[13] Indeed, **L** comprises a set of rules which define a sound and complete logic of high probability (see [Adams, 66] and [Pearl and Geffner, 87]). This probabilistic interpretation, however, is not essential; Kraus *et al.* [88] have developed a system with equivalent power within a preferential semantics setting.

[14] The term "cumulativity" has apparently been coined by Makinson [89]. See also Kraus *et al.* [88].

[15] Just consider a theory with defaults 'if A then B', 'if A and B then C' and 'if C then ¬B'. It is possible to verify in such a theory that both C and B follow from A. B, on the other, does not follow from A and C.

Acknowledgments

I would like to thank Kit Fine and Judea Pearl for insightful discussions. Special thanks to Bill Dolan for his invaluable help with the presentation.

References

[Adams, 66] E.Adams. Probability and the logic of conditionals. In *Aspects of Inductive Logic*, J. Hintikka and P. Suppes (Eds), North Holland Publishing Company, Amsterdam, 1966.

[Apt *et al.*, 88] K. Apt, H. Blair and A. Walker. Towards a theory of declarative knowledge. In *Foundations of Deductive Databases and Logic Programming*, J. Minker (Ed), Morgan Kaufmann, Los Altos, 1988, 89-148.

[Delgrande, 87] J. Delgrande. An approach to default reasoning based on a first-order conditional logic. *Proceedings AAAI-87*, Seattle, 1987, 340-345.

[Etherington, 88] D. Etherington. *Reasoning with Incomplete Information.* Pitman, London, 1988.

[Fine, 88] K. Fine. The justification of negation as failure. Dept. of Philosophy, UCLA, 1988. To appear in *Proceedings of 8th International Congress of Logic Methodology and Philosophy of Science*, North Holland, 1989.

[Geffner and Pearl, 87] H. Geffner and J. Pearl. A framework for reasoning with defaults. TR-94b, Cognitive Systems Lab., October 1987, UCLA. Also in *1988 Proceedings of the Society for Exact Philosophy*, to appear.

[Geffner, 88] H. Geffner. On the logic of defaults. *Proceedings of the AAAI-88*, St. Paul, Minnesota, 449-454.

[Geffner, 89] H. Geffner. *Explorations in non-monotonic reasoning.* Forthcoming dissertation, Computer Science Dept., UCLA.

[Gelfond and Lifschitz, 88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. *Proceedings 1988 Symposium on Logic Programming*, MIT Press, Cambridge, Mass., 1988, 1070-1080.

[Hanks and McDermott, 86] S. Hanks and D. McDermott. Default reasoning, non-monotonic logics, and the frame problem. *Proceedings AAAI-86*, Philadelphia, 1986, 328-333.

[Harman, 86] G. Harman. *Change in View.* MIT Press, Cambridge, MA, 1986

[Haugh, 87] B. Haugh. Simple causal minimizations for temporal persistence and projection. *Proceedings of the AAAI-87*, Seattle, Washington, 218-223.

[Haugh, 88] B. Haugh. Tractable theories of multiple defeasible inheritance in ordinary non-monotonic logics. *Proceedings AAAI-88*, St. Paul, Minnesota, 421-426.

[Horty *et al.*, 87] J. Horty, R. Thomason and D. Touretzky. A skeptical theory of inheritance. *Proceedings AAAI-87*, Seattle, Washington, 358-363.

[Kraus *et al.*, 88] S. Kraus, D. Lehmann and M. Magidor. Preferential models and cumulative logics. Dept. of Computer Science, Hebrew University, Jerusalem 91904, Israel, August 1988.

[Lifschitz, 85] V. Lifschitz. Computing circumscription. *Proceedings IJCAI-85*, Los Angeles, California, 121-127.

[Lifschitz, 87] V. Lifschitz. Formal theories of action. *Proceedings of the 1987 Workshop on the Frame Problem in AI*, Kansas, 1987, pp. 35-57.

[Loui, 87b] R. Loui. Defeat among arguments: a system of defeasible inference. *Computational Intelligence*, 3, 1987.

[Makinson, 89] D. Makinson. General theory of cumulative inference. *Proceedings of the Second International Workshop on Non-Monotonic Reasoning*, Springer Lecture Notes on Computer Science, January 1989.

[McCarthy, 80] J. McCarthy. Circumscription–A form of non-monotonic reasoning. *Artificial Intelligence*, 13, 1980, 27-39.

[McCarthy, 86] J. McCarhty. Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence*, 28, 1986, 89-116.

[McDermott, 82] D. McDermott. Non-monotonic logic II: non-monotonic modal theories. *JACM*, 29, 1982, 33-57.

[Morgenstern and Stein, 88] L. Morgenstern and L. Stein. Why things go wrong: a formal theory of causal reasoning. *Proceedings AAAI-88*, St. Paul, Minnesota, 1988, 518-523.

[Pearl, 88] J. Pearl. Embracing causality in default reasoning. *Artificial Intelligence*, 35, 1988, 259-271.

[Pearl and Geffner, 88] J. Pearl and H. Geffner. Probabilistic semantics for a subset of default reasoning. TR-93-III, Cognitive Systems Lab., UCLA, March 1988. Also in J. Pearl. *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, Los Altos, 1988, Ch. 10.

[Perlis, 88] D. Perlis. Commonsense set theory. In *Meta-Level Architectures and Reflection*, P. Maes and D. Nardi (Eds), Elsevier Science Publishers, 1988.

[Poole, 87] D. Poole. Defaults and conjectures: hypothetical reasoning for explanation and prediction. Report CS-87-4, University Waterloo, October 1987.

[Przymusinski, 88] T. Przymusinski. On the declarative semantics of stratified deductive databases and logic programs. In *Foundations of Deductive Databases and Logic Programming*, J. Minker (Ed), Morgan Kaufmann, Los Altos, 1988, 193-216.

[Reiter, 80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13, 1980, 81-132.

[Reiter, 80b] R. Reiter. Equality and domain closure in first order databases. *JACM*, 27, 1980, 235-249.

[Reiter, 87] R. Reiter. Non-monotonic reasoning. *Annual Review of Computer Science*, 2, 1987, 147-186.

[Sandewal, 88] E. Sandewal. The semantics of non-monotonic entailment defined using partial interpretations. Technical Report LiTH-IDA-R-88-31, Dept. of Computer and Information Science, Linköping University, Sweden.

[Selman and Kautz, 88] B. Selman and H. Kautz. The complexity of model-preference default theories. Proceedings CSCSI-88, Edmonton, Alberta, June 1988, 102-109.

[Shepherson, 88] J. Shepherson. Negation in logic programming. In *Foundations of Deductive Databases and Logic Programming*, J. Minker (Ed), Morgan Kaufmann, Los Altos, 1988, 19-88.

[Shoham, 88] Y. Shoham. *Reasoning about Change.* MIT Press, Cambridge, Mass., 1988.

[Touretzky *et al.*, 86] D. S. Touretzky. *The Mathematics of Inheritance Systems.* Morgan Kaufmann Publishers, Los Altos, CA, 1986.

[Van Gelder *et al.*, 88] A. Van Gelder, K. Ross and J. S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. *Proceedings Seventh Symp. on Principles of Database Systems*, 1988, 221-230.

Induction As Nonmonotonic Inference

Nicolas Helft
ICOT
1-4-28, Mita, Minato-ku, Tokyo 108, Japan
helft@icot.jp *

Abstract

This paper introduces a novel approach to similarity-based inductive reasoning. Induction is defined as inference in a nonmonotonic logic; this approach contrasts with the classical approach that consists of adding formulae to a theory in order to deduce other formulae. We point out problems arising in this setting and show how they are solved within our framework. Given a set of formulae Δ, we define the set Γ of inductive generalizations of Δ, and derive several of its properties.

1 Introduction

This paper introduces a novel approach to empirical (similarity-based) inductive reasoning. The model presented here contrasts with what we call the classical approach to induction: in this approach, a system is presented with information concerning a domain; its task is to infer hypotheses that allow it to "explain" what it observes. From a logical standpoint, what we informally call here "explain" is in fact "deduce". So the task of the system is to add formulae to a theory in order to be able to deduce other formulae. Deduction thus plays a key role in the definition of induction.

This situation can be formalized as follows:

> **Given** some background knowledge Δ and observations Θ, such that $\Delta \not\models \Theta$, **Find** Γ (called *generalizations of* Θ *with respect to* Δ) such that $\Delta \cup \Gamma \models \Theta$. (1)

(Although the problem is not always expressed in logical terms, it is always equivalent to this formulation. See for example the book by Genesereth and Nilsson; we omit additional details that are not relevant here.)

Now, this is certainly a satisfactory model of induction in the framework of scientific, rigourous thinking; but is does not seem to mirror accurately induction as the kind of ubiquitous reasoning of everyday life. For example, upon observing a number of birds and their ability to fly, people might generate the rule that *all birds fly* simply as a conclusion of the observations, grounded on their similarities, rather than as an explanation of the fact that, for example, Tweety flies knowing that it is a bird. No deductive step is involved here, so there is no reason for deduction playing such an important role in the definition of induction.

Contrasting with this, we argue that induction is a process of "jumping to conclusions" in the presence of partial information and thus a kind of inference under uncertainity. Predictably enough, it shares a basic property with certain kinds of default inference: induction assumes that the similarities between the observed data are representative of the rules governing them (we subsequently call it the *similarity-assumption*). This assumption is like the one underlying default reasoning in that a priority is given to the information present in the database. In both cases, some form of "closing-off" the world is needed. However, there is a difference between these: loosely speaking, while in default reasoning the assumption is "what you are not told is false", in similarity-based induction, it is "what you are not told looks like what you are told".

This motivates the approach we introduce here in which, given a set of formulae, we infer other formulae called inductive generalizations of the former. Formally, the problem is

> **Given** a set of formulae Δ (we do not distinguish between background knowledge and observations), **Find** Γ (the *generalizations of* Δ) such that $\Delta \models_{IND} \Gamma$, where $\models_{IND}$ is a certain rule of inference that embodies the assumptions underlying induction.

Γ is supposed to represent all the regularities present in Δ, i.e. all the rules satisfied by its objects. In the machine learning terminology, this is often called "learning by observation and discovery", and is supposed to model a situation in which the learning system receives no assistance from a teacher. However, our aim here is not to model a particular learning situation, but rather to point out problems concerning the way inductive inference is currently formalized in

*This work is based on the author's doctoral research that was done at GRTC, Centre National de la Recherche Scientifique, Marseille, France

AI, and propose a solution for it. The main arguments put forward in this paper thus apply to other learning situations as well.

Basically, we will define the inductive generalizations of a set of formulae as the rules that follow from it by some form of closed-world reasoning. We will use a first-order logic using a "logic database" language, i.e. function-free non-Horn clausal logic. Many definitions and results are taken from Bossu & Siegel's [1985] subimplication, a logic that can be viewed as a particular case of circumscription [McCarthy, 1981] in which every relational symbol has its extension minimalized instead of some, and problems concerning equality are avoided using discriminant interpretations; further relationships between circumscription and subimplication are investigated by Shoham [1987]. Associated with the logic is an attractive language, that of *groundable clauses* (g-clauses for short). Beyond this, the results presented here are independent of any particular application; no bias is assumed other than the one implicit in the representation language.

The emphasis in this paper is on showing the advantages of such an approach to induction. We do not address basic problems such as relevance of the representation language, dealing with noisy information, or learning disjunctive expressions.

The next section discusses problems arising with the formulation (1); section 3 introduces the main definitions; section 4 provides an example; in section 5 we prove some properties of generalizations; we then discuss the motivations behind the definitions and survey related work.

2 Problems with the classical approach

In this section we point out some problems arising with the classical formulation (1). Consider the simple example in which we know Tweety is a bird, and we suddenly observe that he is able to fly. Using the notation of (1), this is expressed:

$$\Delta = \{bird(Tweety)\},$$

and

$$\Theta = \{flies(Tweety)\}$$

We then look for explanations to this fact, i.e. formulae Γ satisfying (1).

$$\Gamma_1 = \{\forall x\, bird(x) \supset flies(x)\}$$

is such a formula, but so are the following:

$$\Gamma_2 = \{\forall x \forall y\, bird(x) \supset flies(y)\}$$
$$\Gamma_3 = \{\forall x\, flies(x)\}$$
$$\Gamma_4 = \{\forall x\, bird(Tweety) \supset flies(x)\}$$

and there are many more. Γ_1 is clearly the intended generalization, but there is no way to prefer it to the others using the definition (1). The solution usually adopted is to introduce additional information in the system, called *bias* in the machine learning literature, a popular one being to favor the more specific formulae over the more general ones (Genesereth & Nilsson [1986] call it the *model maximization* heuristic).

Unfortunately, this doesn't come even close to providing a satisfactory answer: first of all, there are still many solutions, as there can be generalizations incomparable w.r.t. this order, i.e. two formulae neither of which has its models included in the others'.

But the most serious problem is that no reasonable justification exists for choosing such formulae. Of course some form of control is needed to avoid overgeneralization, but the model maximization heuristic performs poorly at this task. Evidence for this is that it does not work in any but the most trivial representation languages: for example, for first-order clausal logic with functional symbols, Buntine [1988] shows that no such most-specific generalization exists (i.e. there is an infinite set of formulae satisfying (1), ordered according to model-inclusion). Although this is a pathological case, he also reports on realistic situations in which these more specific formulae can be clauses of several dozen literals. Niblett [1988] makes a detailed analysis of generalization in first-order logic and shows additional problems arising with most-specific generalizations. Much work has been done on trying to relax the model maximization rule and look for domain-independent information that can help to choose more general formulas [Kodratoff & Ganascia, 1986, Vrain, 1989].

Of course several alternatives exist. For example, use "negative evidence" (e.g. counter-examples of a concept), and look for the version space [Mitchell, 1982] of possible solutions; or introduce domain-dependent bias into the system.

But basically problems remain. We argue that these problems are essentially connected with the formulation (1), and thus cannot be solved within it. The reason is that the fundamental similarity-assumption that grounds inductive inference comes down to preferring some models of the input information to others; the "good" generalizations correspond to the preferred models, and this cannot be captured within the framework (1). We will show that in our setting, from

$$\Delta = \{bird(Tweety), flies(Tweety)\},$$

Γ_1 is the only of the above formulae that can be inferred.

3 Generalizations

This section introduces the main definitions. Given a set of formulae Δ, we define $\Gamma(\Delta)$ [1], the set of generalizations of Δ. This definition aims to capture the intuition underlying inductive reasoning. We will then

[1] We will simply write Γ when there is no confusion.

derive some properties concerning the syntax of formulae in Γ.

We will use a subset of clausal logic and consider *discriminant* interpretations. A property of these is to interpret different ground terms by different elements of the domain (this is equivalent to making the unique-names assumption). So we will identify an interpretation with a set, that of the ground atoms to wich it assigns the value true. Note however that we do not consider Herbrand interpretations, neither do we make the domain closure assumption, so the domain will be infinite, even in the case of a finite set of constants in the initial set and no function symbols.

We consider minimal models in which the extension of every relational symbol is minimalized, i.e., M is a minimal model of a set of formulae if for no other model M', $M' \subseteq M$.

But inductive reasoning occurs over a finite set of objects. We thus need to represent the initial set with a class of formulas for which minimal models always exist; moreover, we want these minimal models to be finite. Here is a class of formulae that has such properties:

Definition: A *groundable clause* (*g-clause* for short), is a clause that satisfies the following properties:

1. its function symbols are constants.
2. every variable that appears in a positive literal also appears in a negative one.

For example, $p(x,y) \supset q(y)$ is a g-clause, while $p(x,y) \supset q(z)$ is not.

The expected properties (proofs in [Bossu & Siegel, 1985]) are the following:

Proposition:

1. Every set of clauses has a minimal model.
2. A set of g-clauses has a finite set of finite minimal models.

The definition of generalization will be in two parts: we will first define the *value* of a clause w.r.t. an interpretation, and then define the generalizations.

We first need for the generalizations to verify a technical condition.

Definition: A clause $P \supset Q$ [2] is *injective* over a set of ground atomic formulae A, whenever there exists a substitution σ, mapping the literals of P onto elements of A, such that for every pair of variables x, y of P, $\sigma(x) \neq \sigma(y)$.

For example, if

$$A = \{hand(1, clubs, clubs), wins(1), hand(2, spade, spade), wins(2)\}$$

then the clause

$$hand(x,y,z) \supset wins(x)$$

is not injective over A because both

$$\sigma_1 = \{x/1, y/clubs, z/clubs\}$$

and

$$\sigma_2 = \{x/2, y/spade, z/spade\}$$

assign the same value to y and z. Of course

$$hand(x,y,y) \supset wins(x)$$

is injective over A.

The generalizations will have to satisfy an injectivity condition over the set of atomic formulas that can be deduced from the original set. There are two reasons for this: firstly, it avoids the introduction of "unnecessary" variables in the generalization; secondly, we will show it is a necessary condition to prove an important property: the set of generalizations is finite.

Definition:

1. Let M be an interpretation and $\phi = P \supset Q$ a clause. The *value of ϕ in M*, denoted $Val(\phi, M)$, is defined as follows:
   1 if $M \models \phi$, $M \models P_i$, a ground instance of P, and P is injective over M. [3]
   0 otherwise.
2. Let Δ be a set of formulae and ϕ a clause. The *value of ϕ in Δ*, denoted $Val(\phi, \Delta)$, is:
   1 if $Val(\phi, M) = 1$ for every minimal model M of Δ.
   0 if $Val(\phi, M) = 0$ for every minimal model M of Δ.
   $\frac{1}{2}$ otherwise.

Example: Let Δ be

$deputy(tom) \vee senator(tom)$
$deputy(x) \supset corrupt(x)$
$senator(x) \supset corrupt(x)$
$rich(tom)$ $rich(bill)$

Δ has two minimal models, M_1 and M_2 that assign true to the following formulae:

$$M1 = \{deputy(tom), corrupt(tom), rich(tom), rich(bill)\}$$

$$M2 = \{senator(tom), corrupt(tom), rich(tom), rich(bill)\}$$

Let
$\phi_1 = deputy(x) \supset rich(x)$
$\phi_2 = corrupt(x) \supset rich(x)$
$\phi_3 = rich(x) \supset corrupt(x)$

[2] Notation $P \supset Q$ means $P = p_1 \wedge \ldots \wedge p_n$, the *premise* and $Q = q_1 \vee \ldots \vee q_m$, the *conclusion*. Alternatively, we will write $\neg P \vee Q$.

[3] As in [Shoham, 1987], if M is an interpretation, $M \models \phi$ means M satisfies ϕ

Then

$$Val(\phi_1, M1) = 1, Val(\phi_1, M2) = 0 : Val(\phi_1, \Delta) = \frac{1}{2}$$

$$Val(\phi_2, M1) = 1, Val(\phi_2, M2) = 1 : Val(\phi_2, \Delta) = 1$$

$$Val(\phi_3, M1) = 0, Val(\phi_3, M2) = 0 : Val(\phi_3, \Delta) = 0$$

So a clause $P \supset Q$ has value 1 in an interpretation M whenever it is satisfied by it (i.e. if $M \models P$ then $M \models Q$), it is not trivially satisfied by it (there is a ground instance P_i of P such that $M \models P_i$), and P does not introduce unnecessary variables as discussed above.

Moreover, we want to take into account disjunctive information, i.e. distinguish completely irrelevant facts from facts that cannot be deduced from the original set but appear in a disjunctive formula that can be deduced from it. Those facts will be true in some minimal model of Δ, but not in all.

But the most important point about the definition is the use of minimal models. This is discussed in section 6.

Generalizations are now defined as follows:

Definition: Let Δ be a finite set of g-clauses.

1. The set of *strong generalizations* of Δ, denoted $\Gamma_S(\Delta)$, is the set of clauses ϕ that satisfy:
   (a) $Val(\phi, \Delta) = 1$
   (b) $\Delta \not\models \phi$
   (c) For any clause ϕ' of $\Gamma_S(\Delta)$, $\phi' \models \phi$ entails $\phi \models \phi'$.
2. The *weak generalizations* of Δ, $\Gamma_W(\Delta)$,are defined likewise, with the first condition replaced by
   (a) $Val(\phi, \Delta) = \frac{1}{2}$
3. The *generalizations* of Δ, $\Gamma(\Delta) = \Gamma_S(\Delta) \cup \Gamma_W(\Delta)$

Condition (b) simply discards formulae in the deductive closure of Δ: we do not want to learn what we "already knew". Condition (c) "cleans up" Γ under subsumption [4]: we do not want Γ to include both $p \supset q$ and $p \wedge r \supset q$. This is not only for aesthetic reasons: it is a necessary condition to prove some of the properties listed in section 5.

4 Example

Let Δ be the following set of formulae:

$expensive - car(rolls - royce)$
$expensive - car(mercedes - benz)$
$drives(x, y) \wedge expensive - car(y) \supset rich(x)$
$deputy(tom)$
$drives(tom, rolls - royce) \vee drives(tom, mercedes - benz)$
$deputy(bill)$
$rich(bill)$
$drives(bill, mercedes - benz)$
$friend(bill, bob)$
$drives(bob, mercedes - benz)$

$\Gamma_S(\Delta)$ then contains the following clauses:
$deputy(x) \supset rich(x)$
$deputy(x) \wedge friend(x, y) \supset rich(y)$
$rich(x) \supset drives(x, rolls - royce) \vee drives(x, mercedes - benz)$

Γ_W contains
$rich(x) \supset drives(x, rolls - royce)$
$rich(x) \supset drives(x, mercedes - benz)$

Δ has two minimal discriminant models. They can be associated with the sets of ground atomic formulae that can be deduced from Δ, to which we add $drives(tom, rolls - royce)$ in one model, and $drives(tom, mercedes - benz)$ in the other.

The formulae in Γ_S are true in both models; those in Γ_W in only one of them.

These formulae thus follow from closed-world reasoning over Δ, i.e. they are true on the objects present in Δ.

Additionally, the following clauses are true in both minimal models of Δ, but there are not generalizations because they fail to satisfy some of the conditions required:

$drives(x, rolls - royce) \supset rich(x)$: can be deduced from Δ

$drives(x, volkswagen) \supset rich(x)$: trivially true in both models: no instance of its premise is satisfied in any model (i.e. nobody drives volkswagen)

$deputy(x) \wedge drives(x, rolls - royce) \supset rich(x)$: subsumed by a formula in Γ_S.

5 Properties of Γ

We now give some results concerning the set of generalizations of a theory. These properties refer to the syntax of such generalizations, and they are consequences of the definition.

We would like to point out the interest of such an approach compared with the traditional one in machine learning. This consists of imposing some *a priori* syntactical restrictions (the learning bias) to the formulae to be learnt. Contrasting with this, our definition did not make any reference to syntax. However, formulas like $deputy(tom)$ (a ground formula), $\neg deputy(x)$

[4] Without function symbols as here, semantic entailment between clauses is equivalent to subsumption: $\phi \models \psi$ iff $\phi\sigma \subseteq \psi$ for some substitution σ. This is not the case if the language contains function symbols.

(a negative formula), or $deputy(x) \wedge friend(y,z) \supset rich(w)$ (which is not really meaningful), would not be intuitively acceptable as generalizations, as they can hardly represent interesting regularities present in the original set.

We now show that this will never occur, as the generalizations satisfy the following properties (proofs are given in the appendix):

Proposition 1: Γ contains only g-clauses.

Proposition 2: There are neither positive nor negative clauses in Γ. [5]

The following definition characterizes in a very natural way formulas that are "meaningful".

Definition:

1. The set of *linked literals* of a clause ϕ is the smallest set containing every literal l of ϕ having one of the following properties:
   (a) l is positive.
   (b) l shares a variable with a linked literal of ϕ.
2. A clause is linked if all of its literals are linked.

Loosely speaking, a link in a clause indicates a "path" between every negative literal and a positive one, through its variables. For example,

$$p(x) \wedge q(x,y) \supset r(y,z)$$

is linked, as $p(x)$ is linked to $q(x,y)$ which is linked to $r(y,z)$,but

$$p(x) \wedge q(y) \supset r(y)$$

is not, as $p(x)$ does not share a variable with a linked literal.

Note that, as this example shows, linked clauses are not necessarily g-clauses, and vice versa.

This definition is motivated by the following:

Proposition 3: Every clause of Γ is linked.

With these results it is easy to prove:

Corollary 4:

1. No clause of Γ contains a negative ground literal.
2. No Horn clause of Γ contains ground literals.

Altogether these results give a strong syntactic characterization of formulae in Γ.

Finally we prove:

Proposition 5: Γ is finite.

We motivate here the injectivity property. Consider the following set of clauses:

$$\phi_1 = p(x_0,x_1) \wedge p(x_1,x_0) \supset q(x_0)$$
$$\phi_2 = p(x_0,x_1) \wedge p(x_1,x_2) \wedge p(x_2,x_0) \supset q(x_0)$$
$$\ldots$$
$$\phi_k = p(x_0,x_1) \wedge \ldots \wedge p(x_k,x_0) \supset q(x_0)$$

and the set

$$\Delta = \{p(a,a), q(a)\}.$$

The above clauses are are unordered w.r.t. model-inclusion, i.e. there is no pair of clauses ϕ_i and ϕ_j such that $\phi_i \models \phi_j$.[6] Moreover, they are all true in the only minimal model of Δ. Fortunately, none of these clauses satisfy the injectivity condition: the substitutions σ grounding its premises are such that $\sigma(x_i) = \sigma(x_j)$, for every pair of variables x_i, x_j of the clause, so none of these is a generalization of Δ. Of course $\phi_0 = p(x_0,x_0) \supset q(x_0)$ is such a generalization. Note that without the injectivity, ϕ_0 would not be in Γ, as it is subsumed by ϕ_1.

Plotkin [1970] also shows similar infinite sets of fuction-free clauses $\{\phi_1, \phi_2, \ldots\}$ ordered in a specific-to-general direction: $\phi_{i+1} \models \phi_i$. If, as in the example above, all these are true in the minimal model of some set of formulae, no generalization would existe as any clause would be subsumed by the next one.

6 Discussion and Comparison With Related Work

6.1 On the use of minimal models

In this section we explain the motivations for the definitions given so far; while doing so we examine alternatives to it and survey related work.

Given a set of formulae Δ, we want to produce all the "hidden" laws in it, i.e. all the rules verified by the objects in Δ. Within a first-order logic, these will then have the form $\forall X\ P \supset Q$. The problem is then to define the weakest conditions that constitute enough evidence to support such a rule.

These weakest conditions are:

- Δ contains an instance of P and Q.
- Δ does not contain an instance of P and $\neg Q$.

Such an approach is taken by [Delgrande, 1985]. In this situation, positive and negative information play a symmetric role, and this leads to a well-known problem in inductive logic, the Hempel paradox: a rule $P \supset Q$ being logically equivalent to its contrapositive $\neg P \supset \neg Q$, with the two conditions listed above one can generate rules with counter-intuitive support. The famous example is a white shoe being support for the rule *all crows are black*: it is supposed that we have information with which we can derive that if an

[5] A *positive (negative)* clause is a disjunction of positive (negative) literals.

[6] Recall that without function symbols, model-inclusion is equivalent to subsumption and observe that for no pair of clauses there is a substitution that makes one clause a subset of the other.

object is a shoe it is not a crow, and if it is white it is not black. The two conditions above thus allows us to conjecture that every non-black thing is not a crow, which is equivalent to say that all crows are black. Delgrande's solution is to change the role of negation within the logic to avoid such paradoxes.

Now, this is to give too much importance to negative information. With the concerns of inductive logic from a philosophical perspective, i.e. to find a logical justification for induction, it may be fine. But artificial intelligence is concerned with modelling human thinking, and it seems quite obvious that positive and negative information do not play symmetric roles in inductive reasoning. Evidence for this is that in virtually every machine learning approach to induction, negative information is used to "control" the generalization process, rather than in a "constructive" way (although see [Nicolas, 1988] for a different view).

The approach we have taken here of course gives high priority to positive information. We thus want the set of generalizations to increase or decrease when positive facts are added to the original database. This naturally leads to the use of minimal models in our definition: entailement over minimal models is *monotonic on positive formulas* (if a positive formula is satisfied by the minimal models of a set of formulas, it is satisfied by all its models [Bossu & Siegel, 1985]). So if a new (i.e. non-deductible) positive fact is added to the database, it will alter its minimal models, and thus it may alter Γ. So this monotonicity result tells us that minimal models achieve the required effect; moreover, it seems to be the weakest condition with which this is verified.

Regarding the conditions that constitute sufficient evidence to support a generalization $P \supset Q$, it is easily proved [Helft, 1988] that our definition imposes the following condition:

- For every instances P_i, Q_i of $P \supset Q$, if P_i is in Δ, so is Q_i.

This condition is of course stronger than Delgrande's. For example, from

$$\Delta = \{bird(Tweety), bird(Opus), flies(Tweety)\}$$

he can derive $\forall x\, bird(x) \supset flies(x)$, while we cannot.

What role do negative formulae thus play in our framework? The same as in any closed-world database: they alter the minimal models only when they can be used to produce a clause that subsumes and thus replaces an older one. For example, if we add $\neg p$ to a database containg $p \vee q$, we are able to derive q and thus use it to produce a new generalization.

To end up this comparison, it is interesting to note that from a model-theoretic perspective, Delgrande's approach is equivalent to minimalizing the left-hand side of the rules and maximalizing the right-hand side.

6.2 Computing Generalizations

In [Helft, 1988], we present an algorithm for computing Γ. Of course there serious complexity problems, as we first need to compute the minimal models of a set of g-clauses, which is equivalent to deriving all the positive ground clauses that can be deduced from the original set. From these, generalizations can be obtained performing a general-to-specific search, with techniques similar to those used by Shapiro [1983]. However, all the syntactical restrictions proved in Section 5 (especially the link between the variables) are used here to dramatically reduce the search space, and the algorithm performs reasonably well at this stage.

6.3 Extensions

We restricted the representation language used here to g-clauses mainly for didactical reasons. However, [Bossu & Siegel, 1985] use a richer language, that of *g-formulae*, which are, loosely speaking, equivalent to rules having existentially quantified positive literals. For example,

$$\forall x\, p(x) \supset \exists y(q(y) \wedge r(x,y))$$

As the results on existence and finiteness of minimal models remain valid for these formulas, generalizations can easily be extended to handle these. For example, instead of the two weak generalizations produced with our example of Section 2, we would get

$$\forall x(deputy(x) \supset \exists y(expensive\text{-}car(y) \wedge drives(x,y)))$$

i.e. every deputy drives some expensive car.

7 Conclusion

The main message of this paper has been the following: there are serious problems with the way similarity-based inductive reasoning is currently formalized in AI. Whatever representation formalism is used, the definition is always based on the idea of "inverting deduction". This definition fails to capture the fundamental similarity-assumption that grounds inductive inference, and this is the cause of the many counter-intuitive generalizations produced by it.

We argued that induction is basically a form of closed-world reasoning, quite close to other forms of default reasoning in AI.

We then defined the set of inductive generalizations of a given set as the formulae that are true in the minimal models of the initial set, and satisfy some additional restrictions. Finally some syntactical properties of such generalizations were derived.

Acknowledgments: I wish to thank especially Pierre Siegel who supervised much of the work from which this paper was produced.

A Appendix: Proofs of Properties

Proposition 1: Γ contains only g-clauses.

Proof: Suppose not. Then Γ contains a clause $\phi = P \supset Q$, for which one of the following hold:

1. Q contains a variable that does not appear in P.
   In this case, let M be a minimal model of the initial set; the two following cases are possible:
   (a) There exists a ground instance P_i of P such that $M \models P_i$.
   Then, let x be a variable that appears in Q and not in P; if $M \models \phi$ then $M \models \forall x\, Q(x)$. Let $Q = Q1 \vee Q2$, where $Q1$ are the literals containing x and $Q2$ the rest of the literals of Q. Then $M \not\models \forall x\, Q1$, because M is finite. So $M \models Q2$. So $M \models P \supset Q2$, which subsumes ϕ. So each time a model satisfies such a clause ϕ, it satisfies a clause that subsumes it. So ϕ is not in Γ because of the last condition in the definition of generalizations.
   (b) Such an instance does not exist.
   So for every ground instance P_i of P, $M \not\models P_i$. So $Val(\phi, M) = 0$, and thus ϕ is not in Γ.
2. $\phi = P \supset Q$ contains a function symbol that is not a constant. Call l a literal in which such function symbol appears.
   In this case, if M is a minimal model of the original set, $M \not\models l$ because of [Bossu & Siegel, 1985] Property 3.2.1 which says that if a minimal model of a set of g-clauses Δ satisfies an atomic formula, this atomic formula contains only constants that appear in Δ. Now,
   (a) if l occurs in P, M satisfies no ground instance of P, so ϕ is not a generalization because $Val(\phi, M) = 0$.
   (b) if l occurs in Q, call $\phi' = \phi - \{l\}$. Then if $M \models \phi$, $M \models \phi'$. The conditions for ϕ and ϕ' to be generalizations being the same, ϕ cannot be one because it is subsumed by ϕ'. This concludes the proof.

Proposition 2: There are neither positive nor negative formulas in Γ.

Proof:

1. No positives: a positive formula is true in the minimal models of a set of formulae if and only if it is true in all models, i.e. if it can be deduced from such a set. Condition (b) in the definition of generalizations discards such formulas from Γ.
2. No negatives: if an interpretation M satisfies a negative clause $\neg P$ (i.e. $M \models \forall X \neg P$), it can never satisfy a ground instance of P.

Proposition 3: Every clause of Γ is linked.

Proof: Suppose not, let $\phi = p \wedge P \supset Q$ be such a clause, p being a non-linked literal, and call $\phi' = P \supset Q$.

If ϕ' is in Γ, ϕ is not since it is subsumed by a clause in Γ. Otherwise, one of the following conditions hold:

1. $Val(\phi', \Delta) = 0$.
   So for every minimal model M of Δ, $Val(\phi', M) = 0$. Again, one of the following must hold:
   (a) $M \not\models \phi'$. Then ϕ' has a ground instance $\phi'_i = P'_i \supset Q'_i$ not satisfied by M. Now consider the ground clause $\phi_i = p_i \wedge P'_i \supset Q'_i$, where p_i is some ground instance of literal p, not satisfied by M. (This is always possible, as M is finite). As p is not linked in ϕ, ϕ_i is necessarily a ground instance of ϕ. As M satisfies neither p_i nor ϕ'_i, it doesn't satisfy ϕ_i either. So M does not satisfy *phi* (as it does not satisfy one of its ground instances), and thus *phi* is not a generalization.
   (b) M satisfies no ground instance of P. Then M will not satisfy an instance of $p \wedge P$ either.
   (c) P is not injective. Then neither is $p \wedge P$ because p has no variable in common with P.
2. $\Delta \models \phi'$
   As $\Delta \models \phi'$ and $\phi' \models \phi$, $\Delta \models \phi$, so ϕ is not a generalization.
3. ϕ' is subsumed by a generalization.
   Then the same clause that subsumes ϕ' subsumes ϕ by transitivity, so ϕ is not a generalization.

Corollary 4:

1. No clause of Γ contains a negative ground literal.
2. No Horn clause of Γ contains a ground literal.

Proof:

1. Negative ground literals are not linked (as they contain no variables).
2. A Horn clause has only one positive literal. It cannot be a ground literal, because no negative literal would be linked. As there are no negative ground literals, there are no ground literals at all.

Proposition 5: Γ is finite.

Proof: Γ has a finite number of finite minimal models. We show that any of these can only satisfy a finite number of injective clauses.

Suppose not, and let $\{\phi_1, \phi_2, \ldots\}$ be an infinite set of such clauses. Then it is possible to construct a set $\{\psi_1, \psi_2, \ldots\}$ such that $\psi_i \subseteq \phi_i$ and all the $\{\psi_i\}$ have the same predicate symbol: as the number of predicate symbols is finite, in an infinite set there must be at least one that appears an infinite number of times.

Γ does not contain redundant clauses: in particular, it does not contain clauses equivalent modulo variable-renaming. So it contains an infinite number of variables, as only finitely many non-equivalent clauses can be formed with a finite number of variables. So the $\{\psi_i\}$ are not injective, since it is not possible to map an infinite set (the variables of the ψ_i) into a finite one (the constants appearing in Δ) with an injective function.

References

[Bossu & Siegel, 1985] Genevieve Bossu and Pierre Siegel: Saturation, Nonmonotonic Reasoning, and the Closed-World Assumption. *Artificial Intelligence* 25(1):23-67, 1985.

[Buntine, 1988] Wray Buntine. Generalized Subsumption and Its Application to Induction and Redundancy. *Artificial Intelligence* 36, 149-176. 1988.

[Delgrande, 1985] James P. Delgrande. A Foundational Approach to Conjecture and Knowledge. Ph.D. thesis, University of Toronto. Technical Report CSRI-173. 1985

[Genesereth & Nilsson, 1986] Michael Genesereth and Nils Nilsson. *Logical Foundations of Artificial Intelligence.* Los Altos, CA: Morgan Kaufmann. 1986.

[Helft, 1988] Nicolas Helft. L'Induction en Intelligence Artificielle, Théorie et Algorithmes. Thèse de Doctorat, Université d'Aix-Marseille II, Septembre 1988.

[Kodratoff & Ganascia, 1986] Yves Kodratoff and Jean-Gabriel Ganascia. Improving the Generalization Step in Learning. In Michalski, Mitchell and Carbonell (Eds.): *Machine Learning: An Artificial Intelligence Approach, vol II.* Los Altos, CA: Morgan-Kaufmann. 1986.

[McCarthy, 1981] John McCarthy: Circumscription – A Form of Nonmonotonic Reasoning. *Artificial Intelligence* 13, 27-39. 1980.

[Meltzer, 1970] Bernard Meltzer: The Semantics of Induction and the Possibility of Complete Systems of Inductive Inference. *Artificial Intelligence* 1, 189-192, 1970.

[Mitchell, 1982] Tom Mitchell. Generalization As Search. *Artificial Intelligence* 18, 203-226. 1982.

[Niblett, 1988] Tim Niblett. A Study of Generalization in Logic Programs. EWSL 88: Third European Working Sessions on Learning. Glasgow: The Turing Institute. 1988.

[Nicolas, 1988] Jacques Nicolas. Consistency and Preference Criteria for Generalization Languages Handling Negation and Disjunction. In Y. Kodratoff (Ed.): *Proceedings of the 8^{th} European Conference On Artificial Intelligence*, pp. 402-407. Pitman Publ. 1988.

[Plotkin, 1970] Gordon Plotkin. Automatic Methods of Inductive Inference. Ph.D. thesis, University of Edimburgh. 1970.

[Shapiro, 1983] Ehud Shapiro. *Algorithmic Program Debugging.* MIT Press. 1983.

[Shoham, 1987] Yoav Shoham. *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence.* MIT Press. 1987.

[Vrain, 1989] Christel Vrain. OGUST, A System Which Learns Using Domain Properties Expressed As Theorems. In Kodratoff & Michalski (Eds.): *Machine Learning: An Artificial Intelligence Approach, vol. III.* To Appear in 1989.

Ontological assumptions in knowledge representation*

Graeme Hirst
Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4

Abstract

If knowledge representation formalisms are to be suitable for semantic interpretation of natural language, they must be more adept with representations of existence and non-existence than they presently are. I review the philosophical background, and exhibit some ontological problems for KR. I then look at the shortcomings of current approaches, including several intensional formalisms and the work of Hobbs. The Meinongian theory of Parsons is considered. Lastly, I present a naïve ontology for knowledge representation, identifying about nine distinct kinds of existence.

1 Introduction

Most contemporary logics implicitly or explicitly base the semantics of the quantifiers $\exists$ and $\forall$ on the widely-held ontological assumptions of Russell [1905, 1918] and Quine [1948]. A small but growing number of philosophers (*e.g.*, [Parsons 1980, Routley 1980, Lambert 1983] believe that these assumptions are mistaken,[1] and have proposed various alternatives. In this paper, I will discuss the consequences of the Russell–Quine assumptions for knowledge representation formalisms, and show that an adequate treatment requires a multi-faceted view of existence.

My motivation comes from the KR needs of natural language understanding. As I have argued elsewhere [Hirst 1988b], a KR formalism to be used in an NLU system for unrestricted text must have (at least) the expressive power of natural language (for otherwise it could not be a target language for semantic interpretation). Moreover, natural languages reflect genuine properties of the real world (with different languages possibly highlighting different properties or viewpoints). Thus, KR research may include exhibiting sentences of natural language and considering how their meaning, and the world it reflects, may be adequately represented—where 'adequately' means that the representation permits the same inferences to be drawn as the original sentence. Here, I am concerned with sentences that speak of existence, of non-existence, or of non-existent objects.

2 Three ontological slogans

2.1 Existence is not a predicate

Immanuel Kant, in his *Critique of pure reason* [1787, B.625ff], argued that existence is not a property that may be predicated of an entity the same way that properties like color and species can be.

Kant was responding to an argument by St Anselm of Canterbury [Anselm 1078, II] that purported to demonstrate the existence of God *a priori*: his 'ontological proof'. Anselm's argument was basically this: What we mean by God is, by definition, that entity that is right up the top end of the scale in all desirable properties: the entity that is most wise, most good, and so on. On the scale of existence, clearly actual or necessary existence is better than mere conceptual or possible existence; therefore existence is a defining property of God; therefore God exists.[2] Descartes [1641, V] later took much the same approach: God has all perfections; existence is a perfection; therefore God exists.[3]

*Parts of this paper were written while the author was at the Department of Artificial Intelligence, University of Edinburgh, with the support of a Visiting Fellowship from the U.K. Science and Engineering Research Council. The balance of the work was supported by the Natural Sciences and Engineering Research Council of Canada.

[1] Introducing his work, Parsons says of the Russell–Quine position that "clear progress is rare in philosophy, and I was pleased to have [it as] an example to cite. But as I thought about it more, I became increasingly dissatisfied" (p. xii).

[2] Compare Smullyan's proof [1978, p. 205–206] that unicorns (or anything else you like) exist: To prove that unicorns exist, it suffices to prove the stronger statement that existing unicorns exist. But for existing unicorns to not exist would be a contradiction; therefore existing unicorns exist; therefore unicorns exist.

[3] For the history of the argument, and a discussion of some of the ontological issues mentioned below, see Barnes

Now, being able to define things into existence like this is metaphysically disturbing, and doesn't really seem possible. Thus, Hume [1779, IX] tried to show that it is not possible that an entity exist of necessity, and Kant took the position described above, which is often characterized as "Existence is not a predicate". This position is now widely accepted in philosophy [Plantinga 1967, p. 38]. Nevertheless, while it may have the merit of keeping God off our backs, it raises difficulties in knowledge representation.

What I want to show in this paper is that existence *can* be predicated, but (lest God be found to be an emergent property of our knowledge representations; no *deus ex machina* here!) it is neither a single predicate nor an ordinary kind of predication.

2.2 Everything exists

An adequate treatment of existence in KR formalisms is complicated not only by the problem described above, but also by a related set of difficulties that derive from a position often summarized by the slogan "Everything exists" (*cf* [Quine 1948, p. 1]). That is, there is nothing that doesn't exist, for if it doesn't exist it isn't anything, and statements apparently about non-existents are either incoherent or can be explained away. The development of this approach is due mainly to Russell [1905, 1918] and, later, Quine [1948]. The Russell–Quine position has become so firmly entrenched in twentieth-century Anglo-American philosophy that it is usually accepted without question [Parsons 1980, p. 1–5]. If we take the slogan literally, then even if existence can be predicated of an entity, it is no more than a tautology; no entities don't exist. And to assert non-existence of something would be self-contradictory. As we will see, this position too is problematic for knowledge representation.

To a large degree, the question seems to be nothing more than what the word *exist* does or should mean, and what status is to be assigned to 'non-existent objects'. Quine grants two kinds of existence: concrete, physical existence in the world (the kind that Margaret Thatcher has), and abstract, non-physical existence (the kind that the number 27 has). "Idea[s] in men's heads" [1948, p. 2] are included in one or the other of these categories, and so too, I assume, are events and actions. Clearly, this is a wider definition of existence than the kind that Anselm and Descartes wished to attribute to God. Presumably they intended some divine equivalent of physical existence—able to have causal interaction with the physical world—and would be unhappy with the idea that God existed only in the way the number 27 does. Likewise, Hume and Kant were using the narrower definition, for many mathematical objects obviously exist of necessity (the number 27; the least prime greater than 27). So perhaps existence in this other sense, non-physical existence without causal connection to the world, *could* be a predicate.

[1972].

2.3 There are things that don't exist

Quine's sense of the word *exist* may be wider than Anselm's and Descartes's, but it is still much narrower than that of Meinong [1904], who described his position in an oxymoron: "There are objects of which it is true that there are no such objects" [1904, Levi *et al* translation, p. 83]. For Meinong (like Brentano before him), every thought or idea, such as the idea of a gold mountain, must be 'directed toward' some object, and so all objects of thought have being in some sense, even if not real-world existence. Meinong therefore wanted to give status to objects such as the gold mountain, which is not real, and the round square, which is not even possible, arguing that the gold mountain is just as good an object as Mount Everest, and the fact that it is unreal makes no difference. Note that the question is not about the *concept* or *idea* of the gold mountain and whether that exists; clearly, it does. But when we say that the gold mountain is 1000 metres tall, we aren't just talking about an idea; it is not the idea that is 1000 metres tall but the alleged thing that the idea is of.

Russell pointed out that Meinong's approach got into trouble with objects like the gold mountain that exists—which isn't real even though existence is part of its definition (*cf* footnote 2). It also troubled him that there was any sense in which a contradiction like a round square could exist.[4]

Thus the question to be considered is what, exactly, do quantifiers like $\exists$ and $\forall$ quantify over? If an expression begins with '$\forall x$' or '$\exists x$', then what values may be used or considered for x? Do they include Margaret Thatcher, the number 23, World War II, my putting the cat out last night, the late Alan Turing, the possibility of rain tomorrow, suavity, fear, the set of round squares, the concept of round squares, or Sherlock Holmes? In other words, what is in the universe of quantification? What exists?

3 What exists?

The burden on the everything-exists position is to explain the apparent counterexamples—the entities that don't exist and yet seemingly form part of the population of everyday naïve ontology. In the next subsection, I will list some of the problematic examples from natural language, and in section 3.2 I will show how Russell tries to dissolve the problems.

[4] Parsons [1980, p. 38–42] has argued that a round square is not a contradiction in the same way a non-square square is, and that the former is a good object but not the latter. Such distinctions need not concern us in this paper.

3.1 What doesn't exist?

Things that aren't there: Perhaps the simplest apparent counterexample (one that we will see Russell's answer to in section 3.2) is that we can explicitly speak of non-existence and non-existent things:

(1) There's no one in the bathroom.

(2) The car I need doesn't exist. [spoken after a long and fruitless search for a suitable car] [Williams 1981, p. 37]

(3) There's no such thing as the bogeyman; he doesn't exist, and neither does Margaret Thatcher.

(4) Nadia doesn't own a dog.

(5) Round squares are impossible, gold mountains merely unlikely.

We may also speak of events that don't occur and actions that are not taken:

(6) A complete lack of money has prevented renovation of the rectory.

(7) The workers threatened to hold a strike if their pay claims were not met. The company acceded to the demands, and the strike was averted.

(8) There are no trains to Saginaw on Sundays. [*i.e.*, the event of a train going to Saginaw on a Sunday does not occur.]

(9) Due to maintenance work on the line, the 6:06 to Saginaw will not run on Sunday.

(10) Today's lecture is cancelled.

(11) Nadia refrained from commenting on Ross's new hairstyle.

(12) Ross failed to notice that Nadia had failed to feed the newt.

Existence itself as an object: We can seemingly speak of existence as an object, one that need not exist:

(13) The existence of Pluto was predicted by mathematics and confirmed by observation.

(14) The existence of Vulcan was predicted by mathematics but disproved by observation.

(15) It's a good thing that carnivorous cows don't exist. [*i.e.*, the non-existence of carnivorous cows is a good thing.]

Claims of reality: We can even (untruly, but not incoherently) assert that unreal objects exist:

(16) I saw a gold mountain near the freeway this morning.

(17) Round squares make me seasick—especially the green ones.

(18) Unreal objects exist.

We can also report such beliefs of others without committing ourselves.

(19) Nadia believes that a unicorn named Old Ironsides has been intercepting her mail and stealing the fashion magazines.

Claims of possibility: We can speak of possible objects and events without committing ourselves either to their reality or unreality, and of objects and events whose existence is merely contingent upon other things.

(20) There may be someone in room 23 who can help you.

(21) If you assemble the parts correctly, you will have created a handsome two-metre model of the CN Tower.

(22) It might rain tomorrow.

Existence at other times: We can refer to things that don't now exist, but did or will. We can speak of things now gone:

(23) Alan Turing was a brilliant mathematician.

(24) Last night's dinner was disastrous.

Sometimes, we may or even *must* even use the present tense for things of the past, suggesting that they have some kind of continuing existence:

(25) *(a)* Alan Turing is a celebrated mathematician. [after Barnes [1972, p. 48]]
(b) *Alan Turing was a celebrated mathematician. [in the sense that he continues to be celebrated]

(26) *(a)* Alan Turing is dead.
(b) *Alan Turing was dead.

And we can talk of things to come:

(27) Tomorrow's dinner is going to be delicious.

(28) The baby that Diane is planning to have will surely interfere with her violin lessons.

Fictional and imaginary characters: We can speak of fictional entities and classes as if they really existed:

(29) Sherlock Holmes was the protagonist of many stories by Conan Doyle.

(30) Sherlock Holmes lived in London with his friend, Dr Watson.

(31) Nadia models herself upon Sherlock Holmes.

(32) Dragons don't have fur. [Plantinga 1967, p. 40]

and possibly even

(33) Sherlock Holmes is no longer alive.

3.2 The Russell–Quine ontology

3.2.1 Paraphrases and the theory of descriptions

Russell's approach, his *theory of descriptions* [1905, 1918], was to regard apparent assertions of existence and non-existence as merely paraphrases—in logic or a literal English rendering thereof—of other forms in which the assertion is not actually made. Instead, the offending bits are expressed as variables and quantifiers, and the resulting expression is something that can legitimately be true or false. Thus, *Dragons exist* is a paraphrase of *There is at least one thing that is a dragon*:

(34) $\exists x(dragon(x))$

Since no such x exists, the sentence is false. Likewise, *Dragons don't exist* is a paraphrase of the negation of (34):

(35) $\forall x(\neg dragon(x))$
'For any x, it is not the case that x is a dragon.'

Attempts to assert properties of non-existent objects may be handled in a similar manner:

(36) Dragons like baklava.
$\forall x(dragon(x) \rightarrow likes\text{-}baklava(x))$

This is vacuously true if there are no dragons [Russell 1918, p. 229]; but statements about particular dragons would be false:

(37) My dragon likes baklava.
$\exists x(my\text{-}dragon(x) \wedge likes\text{-}baklava(x))$

This is false because there is no x for which the left-hand side of the conjunction is true. One might instead have used a vacuously true form like that of (36), but the form of (37) reflects Russell's belief that such sentences were false, and also his concerns with definite descriptions (see below).

In the natural language versions of these statements, we have the apparent problem that even to mention dragons seems to give them some sort of existence; to say that *Dragons like baklava* seems to presuppose the existence of the class of dragons. Russell's claim was that on the 'correct' reading—the representations above, or literal English glosses of them—the problem dissolves. The logical forms contain no assertion of the existence of a non-empty class of dragons. Moreover, the predicate *dragon* is itself a complex term, and may be regarded as simply an abbreviation for a description such as

(38) $fire\text{-}breathing(x) \wedge leather\text{-}winged(x) \wedge \ldots$

Definite references may also be paraphrased. Thus:

(39) The builder of Waverley station was a Scot.

$$\exists x(built(Waverley, x) \wedge \forall y(built(Waverley, y) \rightarrow y = x) \wedge Scot(x))$$

'One and only one entity built Waverley station, and that one was a Scot.' [Russell 1905, p. 113–114]

(If the noun phrase being interpreted does not contain sufficient information to uniquely identify the individual, information from context may be added. Thus (39) might also be the representation of simply *The builder is a Scot* if the context made it clear that the builder was that of Waverley station.) A similar treatment upon *The present king of France is bald* shows the sentence to be false, like (37), because there is no entity denoted by *the present king of France*.[5] Quine [1948, p. 7] showed how the method can be extended to include proper names, so that sentences about named fictional entities might be paraphrased:

(40) Sherlock Holmes is smart.

$\exists x(isHolmes(x) \wedge smart(x))$

'There is an x that has the property of being Sherlock Holmes, and x has the further property of being smart.'

Again, the result is a sentence that is false, for there is no x that has the property of being Sherlock Holmes.

3.2.2 Problems with the theory

Paraphrasing in this manner immediately disposes of some of the problems mentioned in section 3.1, but it does so at some cost.

First, all sentences that assert properties of non-existents are false if talking about a single thing and true if talking about a class, so negating such sentences doesn't change their truth value! For example, the negation of (37) is:

(41) My dragon doesn't like baklava.
$\exists x(my\text{-}dragon(x) \wedge \neg likes\text{-}baklava(x))$

This is false for the same reason that (37) is. Likewise, the negation of (36), *Dragons don't like baklava*, is still

[5] The problem here is, of course, presupposition failure—the sentence tries to talk about something that doesn't exist, and does so without any of the "redeeming" characteristics of the sentences about non-existents that were exhibited in section 3.1. Russell's position on presupposition was famously disputed by Strawson [1950], and is no longer generally accepted. Strawson's position is that the presuppositions of a sentence (or, more precisely, of a particular utterance of a sentence) are distinct from its main assertion, and, unlike the main assertion, are unchanged by sentence negation. If a presupposition is false, then the main assertion, or the sentence itself, can be neither true nor false; rather, it has no truth value at all. For a review of current approaches to presupposition, see [Levinson 1983] or [Horton 1987].

A treatment of presupposition *per se* is beyond the scope of the present paper; for that, see [Horton 1987, Horton and Hirst 1988]. I am concerned here rather with the treatment of the entities that may be felicitously presupposed.

true. The underlying problem here, of course, is that English negation and logical negation aren't the same. If we put a '¬' in front of the logical form of (37), we do change its truth value, but that's not what the English word *not* does. In particular, negation in English (and probably in all natural languages) preserves the presuppositions of the original sentence. In the case of (41), alas, it also preserves Russell's erroneous approach to presuppositions (see footnote 5).

A second problem is a technical one in the nature of the paraphrasing task itself: it destroys, quite deliberately, the similarity between the surface form of the sentence and the representation of its meaning. I have argued elsewhere [Hirst 1987, Hirst 1988a] for the virtues of compositional semantic representations in which each element is a direct reflection of a surface constituent of the sentence. While it is not always possible to maintain this, the advantages to be gained from it are such that it is not to be given up lightly.

Third, and most seriously, there are, as we saw earlier, sentences about non-existents for which one's intuition strongly contradicts the theory of descriptions. These include sentences about defining properties of non-existents and sentences in which non-existents seem to have some actual interaction with the real world.

In the first of these classes, we have sentences such as this:

(42) Dragons have a horn in the middle of their foreheads.

For Russell, this is true, though in any ordinary conversation it would be thought of as false. Likewise, we all agree with Russell and Quine about the falsity of (43):

(43) Sherlock Holmes was stupid.

but we disagree about the reason: in ordinary conversation this sentence is taken as false exactly because (40) is taken as true (*cf* [Parsons 1980, p. 37]).

In the second class are sentences like the following assertions of non-existence themselves. While we might accept representations like (35) for the denial of classes, the denial of the existence of specific entities is trickier. Consider again:

(44) Ross cancelled the lecture.

(45) The [threatened] strike was averted by last-minute negotiations.

On Russell's theory, sentences like these must invariably be false, which is clearly wrong. Notice that paraphrase, in the style of sentence (35), doesn't help here, because these sentences are asserting more than just non-existence; they are asserting a causal relationship. The expression *The strike was averted* means that the strike never occurred—it did not exist—and that some specific action by someone prevented its occurrence. And which strike was averted? The particular strike that the workers threatened to hold, which has specific properties of time, cause, participants, and so on that differentiate it from all other real or potential strikes, all properties that could be used when constructing the description in a Russellian paraphrase. But under Russell's view, we cannot truthfully talk about this strike at all, for it does not exist; any sentence that attempts to refer to it will be false. (Note, as before, that we can't get out of this by saying that the reference is to the idea of the strike; it is not the idea that is averted.)

It might be objected here that to say *The strike was averted* is a looseness of the English language, for one can also use an indefinite reference, saying *A strike was averted*; perhaps this is the basic form that should be interpreted [Barry Richards, personal communication]:

(46) Someone caused that there be no strike.
$\exists y(cause(y, \neg\exists x(strike(x))))$

(We shall allow *cause* as a predicate that takes a proposition in its second argument, and which asserts that the entity in the first argument caused the second argument to be true.) The problem with this tack is the need to say exactly what didn't happen. After all, there are a lot of strikes that were not averted; but (46) says there were no strikes at all. Clearly, some identification from the context is necessary: what was averted was a strike by some particular set of workers at some particular time over some particular claim—so we must identify *the* strike in context, bringing us back to where we started.

Another objection could be that the proper paraphrase is *The strike that was planned was averted*, the claim being that the strike does exist, non-physically, like mathematical objects, by virtue of its having been planned. (This would explain why it sounds a bit funny to say *The accident was averted* instead of *An accident was averted* (*cf* above), as accidents aren't planned.) The problem with this is that one cannot avert mathematical objects any more than one can avert ideas. Perhaps what was averted was the physical realization of this non-physical object—in effect, the instantiation of a concept. I will pursue this line in section 4.2 below.

One could also claim that if the strike was planned, it exists as a 'future object'. To examine this, we must consider the role of time. Unfortunately, Russell provides no treatment of existence at times other than the present, but we can speculate on how he would extend his theory to do so.

Let's consider the simpler case first: the past. It is unclear from Russell's account how he would paraphrase, say, *Alan Turing was smart* and *Alan Turing is dead*. That is, would he allow the scope of quantification to include past entities? Doing so would let the first of these sentences be paraphrased like any other,

and the past-tense verb would just be an artifact of the past-ness of Alan Turing himself, not included in the paraphrase:

(47) Alan Turing was smart.
$\exists x(isTuring(x) \wedge smart(x))$

This would then be a true sentence, unlike *Sherlock Holmes was smart.* But trying this for the second sentence:

(48) Alan Turing is dead.
$\exists x(isTuring(x) \wedge dead(x))$

doesn't work, because Turing wasn't dead when he existed, and the verb tense hasn't behaved. At a minimum, we need to add some notion of time points or intervals such that propositions can be true at some times and not others; thus, (48) would be true today, but false in 1945 and 1862—false in 1945 because Turing was still alive, and false in 1862 because he hadn't yet come within the scope of the existential quantifier.

Thus the universe is seen as travelling through time, collecting up entities into its ontology as it proceeds. Once a thing has started to exist, it never stops. This helps represent sentences (47) and (48), but I don't think this view can be pleasing for the everything-exists gang, for the fact remains that Alan Turing does not now exist in the world any more so than the gold mountain does, nor does he seem to exist as a mathematical object. (The idea of Turing continues to exist, but it's not that that's dead.) There doesn't seem to be any good reason why his brief time on earth should give Turing any subsequent ontological advantage over the gold mountain.[6]

These problems may be seen even more clearly if we now consider future entities, such as the strike that the faculty are threatening to hold. We can talk about this just as easily as we can about Alan Turing (albeit with less certainty)—it will be long and nasty, it will cause the university president to resign, it may never happen (!). For Quine, certainly (and presumably for Russell—guilt by association), the strike is merely a 'possible object', to be kept out of one's ontology at all costs (*cf* his arguments against the existence of the 'possible man in the doorway' [Quine 1948]). So now the averted strike is out on two separate counts, each fatal on its own. When it was still a planned strike, it was merely a possible object; after it was averted, it became a past object as well.

But for KR and NLU, this is simply not acceptable. I have shown above that objects like Alan Turing and the averted strike must be able to be represented, quantified over, and reasoned about just as much as Margaret Thatcher. So the Russell–Quine position is inadequate, and we must look for alternatives. This I will do in sections 5 and 6, after first examining the degree to which KR formalisms share the Russell–Quine deficiencies.

4 Existence assumptions in KR formalisms

To what extent are knowledge representation formalisms able to deal adequately with existence and non-existence? The universe of discourse of a system is, of course, circumscribed by what's in its knowledge base; but given that non-existent entities may have to be included (and, in a full NLU system, *must* be included), how does the average formalism behave?

[6] A rejoinder that I shall not take very seriously: Alan Turing does in fact still exist, or at least his soul does, in Heaven or Hell or somewhere like that. On this view, one might say that the best paraphrase for *Alan Turing is dead* is one of these:

(i) Alan Turing's body doesn't exist (or no longer exists).
$\neg\exists x(bodyOfTuring(x))$

(ii) Alan Turing is in the afterlife.
$\exists x \exists y(isTuring(x) \wedge afterlife(y) \wedge in(x, y))$

Form (i) is undoubtedly true, and the truth of form (ii) depends on whether there is an afterlife and if so who's there (issues that I will not solve in this paper).

The value of this particular objection is to draw attention to the cultural bias in the expression of the problem; perhaps we say that Alan Turing *is* dead just because English reflects our long cultural history of belief in a soul and an afterlife. If we are careful to avoid such bias in our language, we will be able to analyze the problem correctly (or so said a large twentieth-century school of philosophy). Notice, for example, that English offers no analogous expressions for the past existence of objects to which we do not (culturally) attribute an afterlife; if my wristwatch has ceased to be, I can say *My wristwatch was destroyed* but not *My wristwatch is destroyed* (and only as a joke or metaphor, *My wristwatch is dead*). Thus when we say that Turing is dead, our paraphrase should be no more than that there is no x such that $isTuring(x)$; and that this statement was false at an earlier time is an implicature of the word *dead.*

I don't think that this argument goes through. There are too many other things we can say about entities of the past that seem to presume their continued existence:

(iii) Alan Turing {is | *was} a celebrated mathematician.

(iv) Nadia models herself upon Alan Turing.

(v) Nadia knows more about NP-completeness than Alan Turing ever did. [Although Turing is referred to in the past tense, the entity *Alan Turing's knowledge of NP-completeness* is available for comparison with an entity, Nadia's knowledge, that exists in the present and did not exist at the time of Turing.]

(vi) Nadia modelled her new sculpture upon my old wristwatch (which was destroyed last year).

(vii) The Flat Earth Society is now disbanded.

For the most part, KR formalisms are Russellian in their approach to ontology. It is a general characteristic of KR formalisms that even when they are declarative they are assertional—that is, to state something is to assert its truth; one cannot say that something is false. One can, of course, assert the negation of falsehoods on those occasions when this yields truth, but there is no concept of the truth value of a statement being independent of the expression of the statement. Likewise, it is a usual assumption in formalisms that to use a term is to assert that it denotes, and, in particular, that it denotes an extant entity. To assert, for example, *cancelled(lecture23, Ross)*, implies for most systems (*e.g.*, KRYPTON [Brachman *et al* 1983] and conceptual graphs [Sowa 1984]) that *lecture23* exists just as much as *Ross* does, even if the expression says that it doesn't.

4.1 Hobbs: Ontological promiscuity

Not all KR formalisms impute existence to denotations of their terms. A simple first-order system in which (ignoring all the philosophical wisdom discussed above) *exists* is a predicate like any other has been proposed by Hobbs [1985] in his paper entitled "Ontological promiscuity". The 'promiscuity' of the title refers not to the Meinong-like inclusion of all non-existent objects, but rather to reification of events and properties as objects;[7] Hobbs's set of objects, over which quantifiers range and in which all variables are assumed to denote, is a Platonic universe, "highly constrained by the way the ... material world is" (p. 63). The formalism is deliberately simple and 'flat', without modals, intensions, or even negation.

In this approach, no object mentioned in a representation is assumed to exist in the real world unless such existence is either explicitly stated or axiomatically derivable. For example, *Ross worships Zeus* is represented as:

(49) $Exist(E) \wedge worship'(E, Ross, Zeus)$

This says that E is a worshipping by Ross of Zeus, and E exists. The predicate *worship'* is transparent in its second argument but not its third. This means that the existence of E implies the existence of Ross, but not that of Zeus. Hobbs shows that with an adaptation of Zalta's system of abstract objects [Zalta 1983], this approach is able to deal with several problems of opaque contexts that are usually thought to require higher-order representations, while at the same time remaining (moderately) faithful to the surface form of the English sentence.

[7] Treating events as objects, in the style of Davidson [1980], is a position that I have adopted in this paper and assumed to be relatively uncontroversial even for supporters of the Quine–Russell position. Treating properties as objects is a separate question somewhat orthogonal to the concerns of the present paper; suffice it to say here that Quine and Russell would not, I think, approve.

Although Hobbs mentions non-existence only briefly, it is clear that by extending his approach we can account for some of the problems mentioned above. Just as transparent argument positions entail existence, we will allow an argument position to be *anti-transparent*, entailing that the object in that position does *not* exist.[8] (Anti-transparent positions are not to be confused with Hobbs's opaque positions, which entail nothing.) We can then represent the prevention of the occurrence of the strike:

(50) The strike was averted.
$strike(s) \wedge \exists x(Exist(E) \wedge avert'(E, x, s))$

It would be stipulated that *avert'* is transparent in its second argument and anti-transparent in its third—that is, the existence of E implies the non-existence of s.

The existence of existence also seems representable. Hobbs has a 'nominalization operator', ', which turns an n-ary predicate into an $(n + 1)$-ary predicate whose first argument is the condition that holds when the base predicate is true of the other arguments. We saw this above with the ternary predicate $worship'(E, Ross, Zeus)$, derived from the binary predicate $worship(Ross, Zeus)$. Since *Exist* is just another predicate, there is nothing to stop us nominalizing it:

(51) The existence of carnivorous cows is predicted by GB theory.

$$Exist'(E_1, carnivorous\text{-}cows) \wedge predict'(E_2, GB\text{-}theory, E_1) \wedge Exist(E_2)$$

'E_1 is the existence of carnivorous cows, E_2 is the prediction of E_1 by GB theory, and E_2 exists (but E_1 might not).'

On the other hand, there is no treatment of fictional objects. Non-existent objects can be mentioned, as we saw in the assertion of *Ross worships Zeus*, but there is nothing that lets us say that Zeus exists in fiction whereas the Giant Cosmic Groundhog (which I just made up) and the averted strike do not. An obvious move is simply to add a predicate *Fictional* to the formalism. Then *worship'* would have the property that its third argument must exist either in the real world (like Nadia, whom Ross also worships) or in fiction (even if only a small fiction in Ross's mind). Hobbs's Platonic universe would now have a tripartite division into the existent, the fictional, and all the rest.[9]

[8] Hobbs explicitly excludes negation from his formalism, but I shall assume it to be added in the usual way.

[9] I will resist the temptation to be side-tracked onto the question of characterizing more precisely what it means to be fictional. However, there is no principled reason I can see for limiting the property to the imaginary entities in published or oral literature; those in lies and untrue thoughts are just as fictional as Sherlock Holmes ever

But so far, this approach doesn't give an adequate treatment of objects like Alan Turing—for simplicity, Hobbs did not include any notion of time in his formalism, so we can't talk about Turing's different statuses at different times. In addition, it seems that Anselm's fallacy is valid in the system. Although Hobbs gives no examples of definitions, it would seem that *Exist* could be used directly or indirectly as a defining characteristic, it being just another predicate. Its direct use in a definition could be prohibited by stipulation; but preventing its indirect use is not possible, as it is a deliberate feature of the system that existence can be axiomatically derived from various assertions—one has to be allowed to define predicates with transparent arguments. Thus, following Descartes's version of the fallacy, one could define the predicate *perfect* to be transparent in its (sole) argument, and then assert that God is, by definition, *perfect*.

4.2 Intensional approaches

Although it was important for Meinong that thoughts and ideas could be directed to non-existent objects, I have said little up to now, except in passing, about ideas, intensions, and concepts. Indeed, both Russell and Hobbs were at pains to avoid the standard Fregean distinction [Frege 1892] between intension and extension (*Sinn* and *Bedeutung*). But even Quine grants ideas a place in his universe (see section 2.2 above); so we now turn to this topic. I will use the terms *concept*, *idea*, and *intension* interchangeably below; the technical differences between them will be unimportant. Likewise, I will conflate *extension* with the *denotation*, *realization*, or *instance* of an idea.

An adequate treatment of concepts as 'first-class objects' has often eluded knowledge representation systems. By a first-class object, I mean here an object that can be referred to as an individual in its own right, used in inference, be a component of other objects, and so on. This would be necessary if we were to act on the suggestion (section 3.2.2 above) that sentence (45) be represented as the prevention of the realization of an instance of the concept of strikes. Now, because concepts are used to define other objects, many systems accord them a special status that precludes their simultaneously acting as ordinary objects or individuals. A typical example is Frail [Charniak *et al* 1983], a language in which concepts are *generic frames*, but inference can be carried out only on *instances* of those frames; it is not possible for a frame to be simultaneously generic and an instance. In Krypton [Brachman *et al* 1983], which makes a careful separation of 'terminological' knowledge (which goes in its 'T-box') and assertions about the world (in its 'A-box'), it is possible to reason with the terminological knowledge, which can be thought of as statements about concepts, but concepts *per se* can still not be reified as first-class individuals.

Languages in which concepts *are* first-class objects include McCarthy's first-order language [McCarthy 1977, McCarthy 1979], Shapiro and colleagues' SNePS [Maida and Shapiro 1982, Shapiro and Rapaport 1987], and Sowa's Conceptual Graphs [Sowa 1984]. Such languages must provide a mechanism to relate objects to the concepts of which they are instances. For example, Sowa's conceptual graphs tie concepts and their extensions together by notational means. Thus `[CAT:*]` represents the concept of cats, and `[CAT:#234]` represents some particular cat (namely, cat number 234). The notation `[CAT:*x]` represents the individual concept of a cat: a single cat, but not any particular known one; the `x` may be thought of as a variable, so that all occurrences of `[CAT:*x]` must refer to the same (unknown) cat, but `[CAT:*y]` may be a different one. The different types may be used interchangeably (with different meaning, of course) in the graph representations that can be built. However, all graphs are implicitly existentially quantified; that is, the ontology is implicitly Russellian.

Likewise, McCarthy's language has both concepts and extensions as entities, not formally distinguished from one another.[10] A function called *denot* maps concepts to the entities, if any, that they denote. (Thus individual concepts such as *John* are mapped to an individual, and presumably generic concepts like *Dog*, not explicitly mentioned by McCarthy, would be mapped to an appropriate set of individuals.) A predicate *Exists* is true of those concepts for which there is a denotation.[11] 'Parallel' predicates may be defined for denotations and concepts. For example, if *ishorse* is a predicate true of horses, then *Ishorse* can be defined as a predicate true of concepts for which *ishorse* is true of their denotations, and possibly also true of some concepts that don't have denotations, such as *Pegasus*.

The SNePS network formalism is of special interest, as Rapaport [1985b] has suggested that Parsons's theory (section 6 below) could give it a formal semantics. In SNePS, *all* entities are intensions, and extensions *per se* are not used. This is because SNePS takes representations to be those of the knowledge of an

was, even if not as widely known. In the strictest interpretation, this means that just mentioning examples of non-existent entities, such as the Giant Cosmic Groundhog above, gives them status as fictional—which is not really what one wants, as then *all* non-existent entities would be fictional.

[10] The typographical distinctions in McCarthy's formulas are for the reader's convenience, and are not part of the theory.

[11] McCarthy's *Exists* is not to be confused with Hobbs's predicate of the same name (section 4.1 above). McCarthy's *Exists* is a predicate true of concepts that have real-world denotations; Hobbs's *Exists* is true of the real-world objects themselves.

agent, rather than of the world directly. The intensions are connected to reality only through the agent's perception.[12] Like McCarthy, Shapiro and colleagues show only individual concepts, such as the node *John* representing the idea of John;[13] I assume that if the agent is to think about the idea of John, it will need a node that represents the idea of the idea, with a *denot*-like arc relating them.

It is interesting to note that, generally speaking, KR formalisms that treat concepts as first-class objects do not formally distinguish them from individuals. (Those that don't do—they have to, in order to discriminate against them.) I don't know of any principled reason for this. Such systems are *weakly intensional* systems, countenancing intensions but not making anything special of them. In contrast, *strongly intensional* systems take intensions to be not just first-class objects but objects of a distinct kind.[14] Montague semantics [Montague 1973] is a good (non-computational) example of a strongly intensional system.

I suspect that a strongly intensional system will be necessary for an ontologically adequate treatment of intensions. McCarthy could use his *denot* function to map intensions to their extensions, but going in the opposite direction requires an operator, as in Montague semantics. The examples of section 3.1 show such operations to be frequently necessary, and the modes of existence to be discussed in section 7 below suggest that a diverse set of operators may be required.

5 Free logics

One solution that has been suggested to the problems of the Russell–Quine approach is the use of *free logics*. A free logic is a logic that makes no assumptions about existence—specifically, a logic that tolerates terms that have no denotation in its universe but never quantifies over such terms.[15] For example, Woodruff's system **UE** [Woodruff 1970] is a free logic with truth-value gaps (*i.e.*, the truth values **t**, **f**, and **u**) and a distinction between assertions of truth and assertions of non-falsity. Non-denoting terms have no interpretation at all, and a predicate need only have truth value **t** or **f** if all its arguments denote. Thus the system is explicitly Strawsonian. In contrast, Schock's free logic [Schock 1968] has only two truth values, and (in the style of Frege) uses the empty set as the 'denotation' of non-denoting terms. Both systems have an 'existence' predicate, which is true just of those terms that denote.

Free logics seem to be an attractive solution in KR to the problems of Russellianism. From an NLU perspective, free logics help avoid Russellian paraphrases, thereby leading to a more compositional semantics. From a KR viewpoint, they are a conceptually easy extension of classical systems; deduction systems already exist for them; and truth-value gaps are already a focus of research in the field (*e.g.*, Patel-Schneider's four-valued logic [Patel-Schneider 1986]). But alas, free logics turn out to have most of the same problems for NLU as Russell's standard logic. Sentences about non-existents need not be false (at least in Woodruff's logic), but (except in a trivial, unhelpful way) they still can't be true.

6 Parsons: Non-existent objects

Hobbs's scheme implicitly countenanced nonexistent objects, but, as we saw, found itself limited because it tried not to make anything special of the notion of existence. Free logics also accept non-existent objects, but try their best to ignore them. We now turn to an approach that doesn't just accept such objects, but whole-heartedly embraces them. The approach is that of Parsons [1980]; it is explicitly motivated by Meinong's ideas (see section 2.3 above).[16]

Parsons defines the set of *nuclear properties* as the set of properties such as being green, being in New Zealand, or being Nadia. Such properties are "ordinary properties" [Parsons 1980, p. 24] that we regularly attribute to individuals, and corresponding to each is a *nuclear predicate* true of individuals that have that property. Nuclear predicates are in contrast to *extra-nuclear predicates*, of which the prime example is *Exists*. Thus, existence is taken as a predicate, but one of a special kind.

Parsons's universe contains only, for any set of nuclear properties, the unique object that has exactly that set of properties. There is an object that is green (and has no other property but that); there is an object that is both green and Nadia; there is even an object

[12] Thus SNePS is free of extensions only for an external observer of the system. The SNePS objects used by a computational agent that employs the formalism (such as Rapaport's CASSIE [Shapiro and Rapaport 1987]) are the concepts in that agent's 'mind', so to the observer they are intensions. To the agent itself, however, they are subjective extensions, identified with its perceptions of reality.

[13] For simplicity, I will ignore Shapiro's careful distinction between nodes and their names.

[14] Both the distinction and the terminology are due to Graeme Ritchie [personal communication].

[15] Hobbs's system (section 4.1 above) is not a free logic. While it makes no assumptions about real-world existence, it does assume that all terms denote something in the Platonic universe, and it quantifies over them.

[16] Rapaport [1981, 1985a] has also presented a Meinong-inspired theory of non-existent objects. Space does not permit discussion of both theories. The main differences between the two are that *(a)* Parsons has two types of predicate, whereas Rapaport has one type that can be applied in two different ways; and *(b)* Parsons has only one type of object, which may or may not exist, whereas Rapaport distinguishes Meinongian objects ('M-objects') from actual objects ('*sein*-correlates' of M-objects).

that is green and Nadia and Margaret Thatcher. But not all these objects exist in the real world—in some cases because they just happen not to, and in other cases because they are not possible. Being possible is another extra-nuclear predicate.

The tricky part is what to do with non-existent objects like the existent golden mountain. It's an object with the properties of goldenness, mountainhood, and existence, but it's not included in the universe as defined above because existence is not a nuclear property. Nonetheless, it must be accounted for, as we can still talk about it, and the account must not entail its existence. So following Meinong, Parsons introduces the concept of *watering down* extra-nuclear properties to a nuclear ones. Thus for Parsons, there is also a nuclear existence property, call it $Exist_n$, and that's what the existent golden mountain has. Watered-down existence says nothing about real, genuine, full-blown extra-nuclear existence, and the existent golden mountain still doesn't have the latter. A similar story can be told about the possible round square; its possibility is merely the watered-down variety.

The watering-down operation on an extra-nuclear predicate creates a nuclear predicate true of a superset of the objects of which the original predicate was true. That is, if a given object has an extra-nuclear predicate true of it, it will have the corresponding watered-down nuclear predicate true of it as well (but not necessarily vice versa). Anything that exists full-strength also exists in a watered-down way; anything that is full-strength-possible is also watered-down-possible. But it's not clear exactly what sort of a thing these watered-down properties can be if they don't really do anything. What exactly is it that the watered-down-existent gold mountain has that the regular gold mountain doesn't? Just, it seems, an abstract attribution that has no effect on anything except in serving to distinguish the two.

Parsons develops a formal language, called $\mathcal{O}$, for talking about this universe. $\mathcal{O}$ is a second-order modal language with belief contexts; quantification is explicitly over all objects in the universe. The language distinguishes the two types of predicates, and the extra-nuclear predicate of existence, denoted $E!$, has special axiomatic properties. The watering-down operation on extra-nuclear predicates is defined. Using Montague-like techniques [Montague 1973], Parsons shows how $\mathcal{O}$ can act as a semantics for a fragment of English, treating sentences such as

(52) The King of France doesn't exist.
$\neg(\iota x)(E!(x) \wedge \mathit{King\text{-}of\text{-}France}(x))[\lambda y E!(y)]$

Roughly, this says that it is not true that there is—in the actual world—a unique x that both is the King of France and exists in the world; if there is indeed no King of France, this formula is true. Also included in the fragment is the sentence *Every good modern chemist knows more about chemical analysis than Sherlock Holmes* (*cf* sentence (v) of footnote 6).

If we are willing to accept Parsons's approach, then a number of our problems are solved. We can talk about Sherlock Holmes and dragons and other fictional objects all we like (Parsons devotes two chapters to fictional objects). We also have Alan Turing available, and, presumably, all future objects. And we have all objects that don't exist, including the strike that was averted and the lecture that was cancelled—that is, we have the objects that have exactly the properties required, with no necessity that they exist. And the existence of God is expressible in $\mathcal{O}$, but is not a theorem.

Parsons's approach is not without problems. (See [Rapaport 1985c] for a detailed critique.) The most obvious, especially for a computational implementation, is the profligate scope of the quantifiers. A free-logic insight that must be retained is that quantification scope must be restrained. Parsons's universe is much too large to quantify over (although Meinong would do so). But there is no single correct constraint on quantification. For example, it would normally be silly to quantify over all the unwritten books, unthought ideas, or unlived lives; but sometimes, one might have to do so. (An unwritten book is surely reified in the sentence *Ross is going to start writing a book.*) In KR systems, this may not be a practical problem, for the size of the universe is limited by the size of the knowledge base anyway, and even within that, searches would normally be further constrained. This is not to say that a knowledge base cannot contain (finite representations of) infinite objects—the set of integers, for example—but a practical system will normally limit itself to the entities it already knows about and won't capriciously start generating new ones just to see what turns up.

Another problem is that while the averted strike and cancelled lecture are available as objects, we can't do everything with them that we would like. $\mathcal{O}$ can say that an existent Ross stands in a *cancelled* relation to a non-existent lecture, but it is not possible, I think, to explicate the meaning of this as Ross *causing* the non-existence; Parsons did not consider such things.

7 Naïve ontology: The ontology of natural language

The real problem with the Russell–Quine position, the free logic approach, and even Parsons's approach is that they equivocate about existence; they speak as if all things that exist exist in the same way. This is clearly not so. Margaret Thatcher exists, and so does the number 27, but they do so in different ways: one is a physical object in the world, while the other has only abstract existence. But even Quine is willing to grant the existence of mathematical entities—and of concepts in general. If we admit these two kinds of

existence, then perhaps we can find even more kinds if we look. And arguments about the nature of one kind need not hold true of the others.

In fact, we can identify about nine different kinds of existence. In doing so, we will follow Meinong, Parsons, and Rapaport in not limiting existence to things in the world, but attributing it to anything that can be spoken of.[17] In this view, everything exists, but not as Quine meant that slogan. In particular, we solve the problems of sentences (44) and (45) by attributing existence (but not physical actuality) to the lecture and the strike concerned. Thus all terms will denote, and all sentences will be about existent objects, in some sense of existence, and will have the potential to be true.

The various kinds of existence are as follows:[18]

- Physical existence in the present real world, with causal interaction. Margaret Thatcher exists this way.
- Physical existence in a past world (with causal interaction therein, and some indirect causal connection to the present world). The late Alan Turing, for example, exists in a world of the past; he doesn't exist now, but nevertheless he *is*, in the present, a celebrated mathematician, and likewise he *is* dead (see section3.2.2 above).
- Abstract, necessary existence, as of mathematical objects such as 27 and the least prime greater than 27.
- A sort of doubly abstract existence, granted to objects such as $\sqrt{-1}$; mathematicians distinguish such 'imaginary' numbers from other numbers for good reason.
- Existence outside a world, but with causal interaction with that world. In most Western religions, this kind of existence is attributed to God; that is, God is not thought to exist merely the way the number 27 does.
- Abstract, contingent existence in the real world. Freedom, suavity, and fear would come into this category.
- Existence as a concept, which is abstract but contingent, such as the concept of Margaret Thatcher, which need not have existed.[19]
- Unactualized existence,[20] as of the baby that Diane wants to have after she graduates, the strike that the faculty would have held if they hadn't got a pay rise, or Margaret Thatcher's brushing her teeth the day after tomorrow. This category includes objects that could become actual in the future, objects in counterfactuals, 'past' objects that never came into being, and perhaps also impossible objects. Again: objects with this sort of existence are distinct from concepts—it is not just the concept of the faculty strike that was averted, it was the strike itself.
- Existence in fiction. This is the sense in which Sherlock Holmes and dragons exist.[21]

My point here is not to argue for exactly this list of types of existence—that is a topic in philosophy, not knowledge representation—but to demonstrate that however many distinct types of existence there are, it's rather more than two. Any KR formalism that is to be adequate to the task of NLU will need to be able to account for them all—that is, it will treat existence as a set of properties, and, given a particular object's mode of existence, draw inferences accordingly.

8 Conclusion

Before closing, a word should be said about possible worlds. It might be suggested that objects such as the averted strike, the gold mountain, or Nadia's baklava-loving dragon *do* have physical existence, just like Margaret Thatcher, but have it in a different possible world. This misses the point. What we want to talk about and represent is one particular world, usually the actual world, and the question is therefore how dragons and averted strikes exist in the particular world of interest. It is insufficient to say merely

[17] The view presented here goes beyond these authors in that it imposes a taxonomy of existence upon their basic ontology.

[18] It should be clear that these kinds of existence can't all be accounted for just by organizing the IS-A hierarchy the right way. It is true that one can, at the top, make a distinction between abstract and concrete entities. But past existence and unactualized existence are certainly orthogonal to the hierarchy of concrete entities.

[19] One may wish to combine this category with the previous one, saying that concepts are not ontologically distinct from other abstract entities like suavity. I will not take a position on this. Alternatively, one might argue that the existence of a concept may be necessary or contingent depending on its extension. That is, the concept of Margaret Thatcher is as contingent as Margaret Thatcher is, but the concept of the least prime greater than 27 is necessary because its extension is. This category of existence would then be split over the three abstract categories above.

[20] I use this horrible term for want of a better one.

[21] "Everyone knows that dragons don't exist. But while this simplistic formulation may satisfy the layman, it does not suffice for the scientific mind ...The brilliant Cerebron, attacking the problem analytically, discovered three distinct kinds of dragon: the mythical, the chimerical, and the purely hypothetical. They were all, one might say, non-existent, but each non-existed in an entirely different way." (Stanisław Lem. "The third sally, or The dragons of probability." *The cyberiad: Fables for the cybernetic age* (Michael Kandel, translator). NY: Avon books, 1976, p. 76.)

that the status of dragons is that they exist in a different possible world, for so, after all, does Margaret Thatcher. That tells us nothing about the difference between dragons and Margaret Thatcher in the world that we *are* representing.

KR formalisms that are to be suitable for NLU must take account of the many different modes of existence that can be spoken of in natural language. Traditional Russellian approaches are inadequate, as are free logics. However, by taking existence to have a variety of modes, and treating it as a property of objects, an adequate approach can be developed.

Acknowledgements

I am grateful to Chrysanne DiMarco, Diane Horton, Andrew Malton, Chris Mellish, Bill Rapaport, Barry Richards, Graeme Ritchie, Stephen Regoczei, Stuart Shapiro, John Sowa, and Nadia Talent for discussions on these matters and comments on earlier drafts of this paper.

References

[Anselm 1078] Anselm. *Proslogion* (M. J. Charlesworth, translator). Oxford: Clarendon Press, 1965.

[Barnes 1972] Jonathan Barnes. *The ontological argument* (New studies in the philosophy of religion). London: Macmillan, 1972.

[Brachman and Levesque 1985] Ronald Jay Brachman and Hector Joseph Levesque (editors). *Readings in knowledge representation.* Los Altos, CA: Morgan Kaufmann, 1985.

[Brachman *et al* 1983] Ronald Jay Brachman, Richard E. Fikes, and Hector Joseph Levesque. "KRYPTON: A functional approach to knowledge representation." Technical report 16, Fairchild Laboratory for Artificial Intelligence Research, May 1983. Reprinted in [Brachman and Levesque 1985], 411–429. An earlier version appeared in *IEEE Computer*, **16**(10), October 1983, 67–73.

[Charniak *et al* 1983] Eugene Charniak, Michael Gavin, and James Hendler. "The Frail/NASL reference manual." Technical report CS-83-06, Department of Computer Science, Brown University, February 1983.

[Davidson 1980] Donald Davidson. *Essays on actions and events.* Oxford: Clarendon Press, 1980.

[Descartes 1641] René Descartes. *Meditations on first philosophy.* In: *The philosophical works of Descartes* (Elizabeth S. Haldane and G. R. T. Ross, translators), volume I. Cambridge University Press, 1911.

[Feigl and Sellars 1949] Herbert Feigl and Wilfrid Sellars (editors). *Readings in philosophical analysis.* New York: Appleton-Century-Croft, 1949.

[Frege 1892] Gottlob Frege. "Über Sinn und Bedeutung." *Zeitschrift für Philosophie und philosophische Kritik*, **100**, 25–50. Reprinted as: "On sense and nominatum" (Herbert Feigl, translator), in [Feigl and Sellars 1949], 85–102. Reprinted as: "On sense and reference" (Max Black, translator), in Peter Thomas Geach and Max Black (editors), *Translations from the philosophical writings of Gottlob Frege*, Oxford: Blackwell, 1952, 56–78.

[Hirst 1987] Graeme Hirst. *Semantic interpretation and the resolution of ambiguity* (Studies in natural language processing). Cambridge University Press.

[Hirst 1988a] Graeme Hirst. "Semantic interpretation and ambiguity." *Artificial intelligence*, **34**(2), March 1988, 131–177.

[Hirst 1988b] Graeme Hirst. "Knowledge representation problems for natural language understanding." In: Harald Trost (editor), *4. Österreichische Artificial-Intelligence-Tagung: Wiener Workshop Wissensbasierte Sprachverarbeitung: Proceedings* (Informatik-Fachberichte 176), Berlin: Springer-Verlag, August 1988.

[Hobbs 1985] Jerry Robert Hobbs. "Ontological promiscuity." *Proceedings, 23rd annual meeting of the Association for Computational Linguistics*, Chicago, June 1985, 61–69.

[Horton 1987] Diane Lynn Horton. *Incorporating agents' beliefs in a model of presupposition.* MSc thesis, Department of Computer Science, University of Toronto; published as technical report CSRI-201, Computer Systems Research Institute, University of Toronto, August 1987.

[Horton and Hirst 1988] Diane Lynn Horton and Graeme Hirst. "Presuppositions as beliefs." *Proceedings, International conference on computational linguistics (COLING-88)*, Budapest, August 1988, 255–260.

[Hume 1779] David Hume. *Dialogues concerning natural religion* (Norman Kemp Smith, editor). Oxford: Clarendon Press, 1935.

[Kant 1787] Immanuel Kant. *Critique of pure reason* (second edition) (Norman Kemp Smith, translator). Revised edition, London: Macmillan, 1933.

[Lambert 1983] Karel Lambert. *Meinong and the principle of independence* (Modern European philosophy). Cambridge University Press, 1983.

[Levinson 1983] Stephen C. Levinson. *Pragmatics* (Cambridge textbooks in linguistics). Cambridge University Press, 1983.

[Maida and Shapiro 1982] Anthony S. Maida and Stuart Charles Shapiro. "Intensional concepts in propositional semantic networks." *Cognitive science*, **6**(4), October–December 1982, 291–330.

[McCarthy 1977] John McCarthy. "Epistemological problems of artificial intelligence." *Proceedings, 5th International Joint Conference on Artificial Intelligence*, Cambridge, Massachusetts, August 1977, 1038–1044.

[McCarthy 1979] John McCarthy. "First order theories of individual concepts and propositions." In: Jean Elizabeth Hayes, Donald Michie, and L. I. Mikulich (editors). *Machine intelligence 9.* Chichester: Ellis Horwood, 1979, 129–147. Reprinted in [Brachman and Levesque 1985], 523–533.

[Meinong 1904] Alexius Meinong. "Über Gegenstandstheorie." In: Alexius Meinong (editor). *Untersuchungen zur Gegenstandstheorie und Psychologie*, Leipzig: Barth, 1904. Reprinted in: Alexius Meinong. *Gesamtausgabe* (Rudolf Haller, and R. Kindinger, editors), Graz: Akademische Druck- und Verlaganstalt, 1969–1978, Volume II, 481–535. In English as: "The theory of objects" (Isaac Levi; D. B. Terrell; and Roderick Milton Chisholm, translators). In: Roderick Milton Chisholm (editor). *Realism and the background of phenomenology* (The library of philosophical movements), Glencoe, IL: The Free Press, 1960, 76–117.

[Montague 1973] Richard Montague. "The proper treatment of quantification in ordinary English." In: Kaarlo Jaakko Juhani Hintikka, Julius Matthew Emil Moravcsik, and Patrick Colonel Suppes (editors). *Approaches to natural language: Proceedings of the 1970 Stanford workshop on grammar and semantics.* Dordrecht: D. Reidel, 1973, 221–242. Reprinted in: Richard Montague. *Formal philosophy: Selected papers of Richard Montague* (Richmond Hunt Thomason, editor). New Haven: Yale University Press, 1974, 247–270.

[Parsons 1980] Terence Parsons. *Nonexistent objects.* New Haven: Yale University Press, 1980.

[Patel-Schneider 1986] Peter Frederick Patel-Schneider. "A four-valued semantics for frame-based description languages." *Proceedings, Fifth national conference on artificial intelligence (AAAI-86)*, Philadelphia, August 1986, 344–348.

[Plantinga 1967] Alvin Plantinga. *God and other minds: A study of the rational justification of belief in God.* Ithaca: Cornell University Press, 1967.

[Quine 1948] Willard Van Orman Quine. "On what there is." *Review of metaphysics*, **1**, 21–38. Reprinted in: *Proceedings of the Aristotelian Society*, supplementary volume 25: *Freedom, language, and reality*, 1951, 216–233. Reprinted in: Willard Van Orman Quine. *From a logical point of view: Logico-philosophical essays.* Harvard University Press, first edition, 1953; second edition, 1961; second edition, revised, 1980, 1–19.

[Rapaport 1981] William J. Rapaport. "How to make the world fit our language: An essay in Meinongian semantics." *Grazer philosophische Studien*, **14**, 1981, 1–21.

[Rapaport 1985a] William J. Rapaport. "Nonexistent objects and epistemological ontology." *Grazer philosophische Studien*, **25/26**, 1985/86, 61–95.

[Rapaport 1985b] William J. Rapaport. "Meinongian semantics for propositional semantic networks." *Proceedings, 23rd annual meeting of the Association for Computational Linguistics*, Chicago, June 1985, 43–48.

[Rapaport 1985c] William J. Rapaport. "To be and not to be" [Review of Parsons 1980]. *Noûs*, **19**(2), June 1985, 255–271.

[Routley 1980] Richard Routley. *Exploring Meinong's jungle and beyond: An investigation of noneism and the theory of items* (interim edition). Departmental monograph 3, Philosophy Department, Research School of Social Sciences, Australian National University, 1980.

[Russell 1905] Bertrand Russell. "On denoting." *Mind*, n.s. **14**, October 1905, 479–493. Reprinted in: Bertrand Russell. *Essays in analysis* (Douglas Lackey, editor), London: George Allen & Unwin, 1973, 103–119. Reprinted in [Feigl and Sellars 1949], 103–115. Reprinted in [Russell 1956], 39–56.

[Russell 1918] Bertrand Russell. "The philosophy of logical atomism." In [Russell 1956], 175–281.

[Russell 1956] Bertrand Russell. *Logic and analysis: Essays 1901–1950* (Robert Charles Marsh, editor), London: George Allen & Unwin, 1956.

[Schock 1968] Rolf Schock. *Logics without existence assumptions.* Stockholm: Almqvist & Wiksell, 1968.

[Shapiro and Rapaport 1987] Stuart Charles Shapiro and William J. Rapaport. "SNePS considered as a fully intensional propositional semantic network." In: Nick Cercone and Gordon McCalla (editors). *The knowledge frontier: Essays in the representation of knowledge* (Symbolic computation series), New York: Springer-Verlag, 1987, 262–315.

[Smullyan 1978] Raymond M. Smullyan. *What is the name of this book?—The riddle of Dracula and other logical puzzles.* Prentice-Hall, 1978.

[Sowa 1984] John F. Sowa. *Conceptual structures: Information processing in mind and machine* (The systems programming series). Reading, MA: Addison-Wesley, 1984.

[Strawson 1950] Peter Frederick Strawson. "On referring." *Mind*, n.s. **59**, 1950, 320–344.

[Williams 1981] Christopher John Fards Williams. *What is existence?* (Clarendon library of logic and philosophy). Oxford: Clarendon Press, 1981.

[Woodruff 1970] Peter W. Woodruff. "Logic and truth value gaps." In: Karel Lambert (editor), *Philosophical problems in logic: Some recent developments* (Synthese library), Dordrecht: D. Reidel, 1970, 121–142.

[Zalta 1983] Edward N. Zalta. *Abstract objects: An introduction to axiomatic metaphysics* (Synthese library 160). Dordrecht: D. Reidel, 1983.

A Framework for Dynamic Representation of Knowledge : A Minimum Principle in Organizing Knowledge Representation

Yoshiteru Ishida
Division of Applied Systems Science
Kyoto University, Kyoto 606 Japan

Abstract

Dynamic memory organization has been proposed by R. C. Schank which allows content addressing in a case base. Case-based reasoning on the dynamic memory proceeds in the following two steps: (1) For the problem solving of the given case, find the nearest cases in the case-base, (2) transform the cases so that the information in the case is applicable to the current case.

In this paper, we first introduce the mechanism to drive the dynamic organization of the memory. In the memory, knowledge is represented by the frame which is structured to maintain the number of slots small by sharing the common attribute-values among cases. An architecture for case-based diagnosis is presented, using the memory organization where diagnostic cases are indexed both by the symptoms and by the causal relation among symptoms. The program CAOS (CAse Operation System) is implemented for the process diagnosis domain.

1 Introduction

Rule-based reasoning is used for most of the knowledge-based systems, however; many other reasoning methods are used in human problem solving. One of such reasoning method, which seems complementary to rule-based reasoning is case-based reasoning. For comparison, consider the problem solving process in judgement. In making judgement, judges use two methods: (1) applying laws to the present case, and (2) transforming the judgement of the similar cases. Roughly, case-based reasoning may correspond to the latter, while rule-based reasoning to the former. Case-based reasoning [Schank 1986] uses analogical mapping from the previous cases to the present cases under consideration. Among many types of analogical reasoning [Winston 1982,Carbonell 1985], case-based reasoning have been studied as a reasoning method on the dynamic memory [Schank 1982].

IPP[Lebowitz 1982] generalizes from the instances of newspaper stories such as terrorism. We will compare this generalization mechanism with ours later(section 2.2). CYRUS[Kolondner 1986] organizes memory by indexing cases by the different features. JUDGE[Bain 1986] uses the case-based reasoning in sentencing in the judgement. CHEF[Hammond 1986] is a case-based planing system applied to cooking. Its dynamic memory indexes the past cooking experiences by the goal they must satisfy and by the problems they must avoid. Then, the indexes are used by reasoner to organize goals for the current case.

We introduce the dynamic representation of knowledge following this line. The structure of human knowledge is dynamically updated whenever the new fact is stored. The structural changes depend upon the abstraction level of the new fact, how it is related to the already structured fact, and how it contradicts or fits to the current structure. Whenever memorizing the new fact we always associate it with the already stored fact having certain relation with the fact. What is related plays an important role when recalling the memorized fact. In realizing such dynamic memory, we pursue a mechanism of the dynamic reconfiguration of the knowledge representation. We use frame representation where the generalization is made by saving as many slots as possible by sharing common slots.

In section 2, the memory organization and general-

ization mechanism adopted here are presented. The problem of making proper genralization is formalized as combinatorial optimization problem. Section 3 presents the architecture for case-based diagnosis utilizing the memory organization. By the program CAOS, how the memory organization is used for case-base of diagnostic cases and how the case operation is selected are discussed. Section 4 states the implementation and shows a sample session of CAOS. Section 5 discusses the impact of introducing dynamic case-base into the knowledge-based systems.

2 Dynamic Structuring of Cases

2.1 Generalization by Saving Slots

In this section, we consider the problem of finding a simple frame representation (each frame expressing a case) structured by inheritance. In order to structure frames in a simple form, generalization having many slots out of many instances must be made. To obtain such generalization, the next problem must be solved. In order to formalize the problem set theoretically, each case is expressed by a set of attribute-value pairs. One slot is used to store one attribute-value pair.

[Minimum Slot Problem]

Given the set of cases $\{S_i\} i = 1..n$ where each case S_i is characterized by the set of attribute-value pairs, find the set S_0 of attribute-value pairs such $\sum_{i=1}^{n} s_i - |S_0|$ ($|S|$ denotes the cardinality of a set S) is the maximal where

$$s_i = \begin{cases} |S_0| & \text{if } S_i \supseteq S_0 \\ 0 & \text{otherwise} \end{cases}$$

S_0 is the generalization from the subclasses S_i ($\supseteq S_0$). $\sum_{i=1}^{n} s_i - |S_0|$ is the number of slots which can be saved by letting S_i such that $S_i \supseteq S_0$ be subclasses of S_0.

This problem is a combinatorial optimization problem [1] which must find the generalized class S_0 satisfying two contradicting requirements: (1) S_0 must contain as many attribute-value pairs as possible, and (2) S_0 must have as many subclasses as possible. The structuring of cases is carried out by iteratively solving this MSP. That is, find S_0 for a given set of cases $\{S_i\} i = 1..n$, and let all the cases S_i such that $S_i \supseteq S_0$ be subclasses of S_0 where all the attribute-value pairs inherited from S_0 are omitted. Then again, carry out this process on the structured case-base iteratively.

We use the notation:

```
([element-name] name=[case-name]
                [attribute-value-pair1]
                [attribute-value-pair2]...)
```

for expressing the case whose name is [case-name] and arbitrary number of attribute-value pairs where each attribute-value pair is expressed as: attribute-name, predicate and value. The name field is not counted as an attribute-value pair.

Let us consider the example of structuring many classes of squares:

```
(fig name=square          corner=4 side=4)
(fig name=lozenge         corner=4 side=4
                            EqualSide=4)
(fig name=trapezoid       corner=4 side=4
                            ParallelSide1=1)
(fig name=parallelogram   corner=4 side=4
            ParallelSide1=1 ParallelSide2=1)
(fig name=rectangle       corner=4 side=4
           ParallelSide1=1 ParallelSide2=1
                              EqualAngle=4)
(fig name=PerfectSquare   corner=4 side=4
          ParallelSide1=1 ParallelSide2=1
                 EqualAngle=4 EqualSide=4)
```

The hierarchical structure obtained by solving the MSP is shown in Fig. 1. The numbers of slots shared by the superclasses are shown. First, S_0 = (fig name=square corner=4 square=4) is chosen as a superclasses of the rest of classes, since placing the rest of classes under this S_0 can save $2 \times 6 - 2 = 10$ slots.

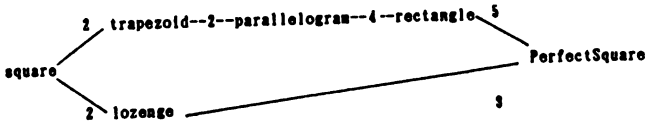


Fig. 1 Hierachcal Structure of the Minimum Slot Representation

Since this reconfiguration process requires huge computations, it is not practical to reconfigure whenever

[1] This problem is NP-complete. Graph theoretically, this MSP is the problem of finding complete subgraph with maximal number of edges for a given bipartite graph.

the new case is stored. Instead, we use the following incremental algorithm which is more efficient, but does not gurantee that the structure is the same as the one obtained by solving MSP.

Before presenting the incremental algorithm, a concept of a distance between related cases C1 and C2 must be defined.

$d(C1, C2) = |C1/C2|/|C1 \cup C2|$

where C1/C2 denotes the symmetric difference of two sets C1 and C2, i.e. $C1/C2 = (C1 - C1 \cap C2) \cup (C2 - C1 \cap C2)$. When three cases, A, B, and C are related as shown in Fig. 2, the case A is called *supercase* of the cases B and C, and cases B and C are called *subcase* of the case A. The case B is called *close case* of C. If there is no relation between two cases, the distance between them is equal to 1.

Fig. 2 Relation among Cases

In the reasoning discussed in section 3, the distance is used to specify the case transformation operation to apply.

[The Incremental Algorithm]

For a given the case,

1. Reminding: Find the nearest case in the case-base in terms of distance defined above.
2. Storing: Insert the given case into the case-base by associating it with the nearest case as follows.
   - If the given case is *super(sub)case* of the nearest case(s), insert the given case as super(sub)class in an appropriate position.
   - If the nearest case is the *close case* of the nearest case(s), insert the case as the subclass of the common superclass if possible, or make the superclass of these two cases and place these two cases as the subclass of the superclass.

2.2 Evaluation of the Generalization

Without considering the causal and hierarchical relation among attributes in each case, erroneous generalization may occur.

Lebowitz avoid this problem by introducing confidence with each new generalization. The confidence is confirmed (disconfirmed) by the positive (negative) evidences against the generalization. The generalized class is regarded as tentative until enough cases which support the generalization enter, otherwise it is removed.

Since we want to keep the memory organization simple and general, we do not facilitate such evaluation mechanism. It seems that the generalized class which saved many slot tends to be correct, for otherwise all the subclasses under the generalized classes includes the erroneous set of attributes. Thus, as long as there are many cases of sufficient varieties in the case-base, the generalization which can save many slots corresponds to the correct generalization.

The structure obtained by iteratively solving MSP tends to be simple, keeping the number of generalized classes small; however the MSP is NP-complete. The structure of cases obtained by *the Incremental Algorithm* is highly dependent on the order of cases input. Thus, another limitation for the memory organization mechanism is that it is sensitive to the order of cases input.

Despite these limitations, generalization mechanism, when worked properly, have the following merits (which will be explained by the examples in the next subsection): (1) It can abstract diagnostic case of single fault out of many cases of multiple faults. (2) It can filter out unrelated and erroneous symptoms in diagnostic case.

2.3 Memory Organization Applied to Diagnostic Cases

Let us consider a simple example of diagnosis of the process shown in Fig.3 [Belenblut and Whitehouse 1973].

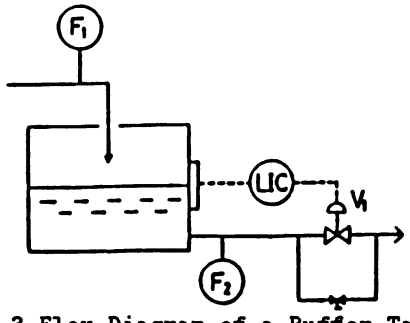


Fig.3 Flow Diagram of a Buffer Tank
[Belenblut and Whitehouse 1973]

The process is a simple buffer tank having inflow F1 (to the tank) and outflow F2 (from the tank) of the liquid. The level L1 of tank is controlled within a certain range by adjusting valve V1. The netflow DF is defined as F1 - F2. Table 1 is a decision table for diagnosis where each column indicates the rule associating syndrome of

sensor pattern with the events on the bottom row. Our problem here is structuring cases of events so that case-based reasoning is possible.

Table 1 Decision table used for diagnosis
[Belenblut and Whitehouse 1973]

Rule	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ΔF	+1	-1	N																	
F1				L	H	N														
L1							L	L	H	H	*									
V1							O	N	N	C	*	O	O	O	N	N	C	C	C	N
F2											L	N	H	L	H	L	N	H	H	N
Event	5	A	1	61	6h	1	A	A	A	A	1	4	4	6h	4	2, 3	5, 61	2, 3	2, 3	1
	Flow Balance		Supply		Control Loop					Leaks and Blockage										

*=else

+1=positive -1=negative N=normal H=high L=low O=open C=close

Events:
(1) Normal operation
(2) Pipe leakage between F2 and V1
(3) V1 by-pass open in error
(4) Blockage in exit line
(5) Leak in tank
(6h) Abnormal throughput high
(61) Abnormal throughput low

(A) Anomally (defined as a physically impossible combination of measure outputs from the point of view of the plant)

2.3.1 Generalization in diagnostic case-base

To see how the generalization mechanism works, consider the next cases given initially,

```
(event name=BlockInF2&V1_15
       fault=block location=F2&V1
       F1=high V1=open F2=low)
(event name=BlockInF2&V1_24
       fault=block location=F2&V1
       F1=low V1=open F2=low)
(event name=BlockInF2&V1_6
       fault=block location=F2&V1
       F1=normal V1=open F2=low)
```

The generalization mechanism create the next superclass out of these three cases.

```
(event name=GEN1
       fault=block location=F2&V1
       V1=open F2=low)
```

Thus, the unrelated attribute F1 is filtered out in this generalized class. Further, when cases of multiple faults are given such as;

```
(event name=BlockInF2&V1&AnormalyInControl_31
       fault=anormaly location=control
       fault2=block location2=F2&V1
       F1=normal V1=normal F2=low L1=high)
(event name=LeakInTank&AnormalyInControl_31
       fault=anormaly location=control
       fault2=block location2=F2&V1
       F1=high V1=normal F2=normal L1=high)
```

Then, the follwoing single fault case is generalized.

```
(event name=GEN2
       fault=anormaly location=control
       V1=normal L1=high)
```

2.3.2 Introducing causal relation in indexing

In diagnosis, two faults can be quite different even if the syndromes of them resemble each other. Thus, using only the cases which has the similar syndromes leads to poor diagnosis. We induce causal relations from syndrome for each case. The causal relation among symptoms indicates the the qualitative direction of the interaction among attribute.

As discussed in analogical reasoning[Winston 1982], causal relations must be preserved in mapping values from known case to the current case. This heuristic is universal one which hold in analogical reasoning of most domains. In fault diagnosis, we introduce the causal relation, which is used for indexing cases.

A base model is needed to generate causal relation from symptoms. The base model, as in Fig. 4, shows all the path from one attribute to the other attributes when the value of the former may cause the value of the latter. A base model is the component dependent knowledge, and hence only one base model is given which are commonly shared by all the cases for the example.

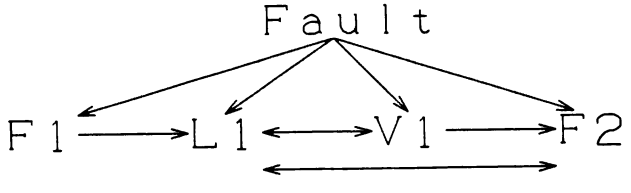


Fig. 4 Base Model for the Buffer Tank Example

There is a causal path from F1 to L1, V1, F2. But no causal path exists from F2 to F1 in this example. The

base model must be given beforehand. With the base model, causal relations are induced as follows. If a case has F1=high and V1=open as its symptoms, then the effect pattern that F1=positive_V1 is generated, since there is a causal path from F1 to V1(positive or negative is determined whether the values of these attributes are same or opposite direction). For example, the following two cases are quite different patterns.

```
(event name=ThroughputHigh_14 type=data
       fault=high location=throughput
       V1=open F2=high)
(event name=ThroughputLow_27 type=data
       fault=low location=throughput
       V1=close F2=low)
```

However, the induced causal relations induced from these two cases are same.

```
(event name=ThroughputLow_16 type=effect
      throughput=positive_V1
      V1=positive_F2)
(event name=ThroughputHigh_18 type=effect
      throughput=positive_V1
      V1=positive_F2)
```

Here after, we will call the syndrome given for eacs case as *data type*, and causal relations among these symptoms as *effect type*.

3 Reasoning on Structured Case-Base

The problem to reason about is; given a new case which may include unknown attribute, infer the unknown attribute based on value of corresponding attribute of the related case. Reasoning on the structured case-base has two steps:

1. Reminding: find the related case in the case-base as in section 2.1.

2. Case operation: transform the nearest case(s) by the operator. Operator is selected how the given case is related with the nearest case.

The reminding process is carried out by the same one as that used in the reconfiguration process stated in the *incremental algorithm*.

We will see the reminding, and case operations in the following subsections in more detail using the example.

3.1 Reminding - Searching the Related Cases

We have already discussed how cases are structured in the case-base. The same process occurs in reminding whenever a new case is entered in the structured case-base. Suppose several cases are already structured in the case-base. In order to use several types of relations in associating the related case, vocabularies used for specifying attribute-values must be related from many view points. The followings are some of them used for the process diagnosis example.

```
(relation name=opposite arg1=high arg2=low)
(relation name=positive arg1=open arg2=high)
(relation name=near     arg1=normal arg2=high)
```

In CAOS, cases are indexed both by *data type* (symptoms) and by the *effect type*(causal relation among symptoms). Reminding (finding the similar cases) is carried out on these two representations.

3.2 Case Operations - Transforming Related Cases

Depending upon the reminded case(s) and its relation to the current case, several candidates for case operations are proposed for the transformation of reminded cases. Case operations are divided into two types: single case operation and multiple case operation. Single case operation uses the information of the only one related case, and multiple case operations uses that of more than two related cases. In this subsection, we will see how these case operations are selected using the information of how the current case is related to the existing case(s).

Further, for the case operation to infer the unknown fault type from the known fault type, these types must be related in the event space.

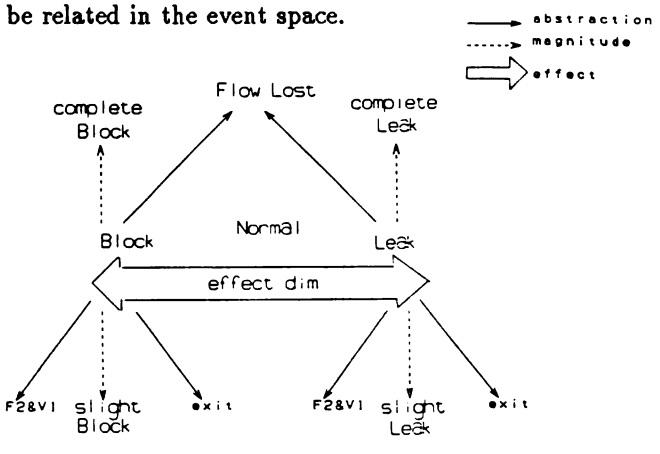


Fig.5 Event Space for Operations

Fig. 5 shows the event space in which the nearest faults related are operated to infer the unknown value of the given falut. These operators and vocabularies are domain dependent.

With the structured case-base, relations among vocabularies specifying values, and event space, case operations can be carried out. The followings are some examples of case operations. These operations are organized so that the strong heuristics must be applied earlier than weak ones by the rule control mechanism.

Sample scenarios of inferring the unknown fault follows as below.

For the given case,

```
(event name=target status=current type=data
       F1=high F2=high L1=high V1=open)
```

suppose the same case has not yet been given to the case-base , however, the next case exists in the case-base.

```
(event name=ThroughputLow_27 type=data
       fault=low location=throughput
       F1=low F2=low L1=low V1=close)
```

As pointed out, this case is the nearest in terms of the *effect type*, although this case is quite different in terms of *data type* . In fact, the distance of *data type* is equalt to 1 whereas the distance of *effect type* is equal to 0.

Fig. 6 is the rule expressed by OPS83 [Forgy 1986] proposing the case operator.

```
rule FaultLocation_propose_OppositeType
{
&g (goal PhaseStatus=current; step=propose;
        ProblemSpace=FaultLocation);
&e (event status=current );
&d (distance from=&e.id; type=effect;
    (@.name=super\/@.name=sub\/@.name=close);
          value<=&g.Edistance);
   (event id=&d.to);
  -(distance from=&d.from; to=&d.to
            type=data;);
  -(operator object=&d.to;
            ProblemSpace=OppositeType;
   (@.PhaseStatus=done\/
    @.PhaseStatus=pending););
   -->
    modify &g (Edistance=0.0; Ddistance=0.0);
    make (operator id=gint ();
                  PhaseStatus=pending;
                  status=proposed;
                  ProblemSpace=OppositeType;
                  priority=5;
                  object=&d.to );
};
```

Fig. 6 OPS83 rule for case operation making the fault type opposite

Only one case operator is selected among candidate case operators by the heuristics and default rule using priority. The control structure of the case operation part uses the control structure of SEAR [van de Brug 1986] modified for OPS83. We will describe implementation of CAOS later.

The English translation of this rule follows:

> Even if the case is not related as to *data type*, if it is related as to *effect type*, then propose the candidate case operator which guess the fault type of the current case as opposite of the related case.

Applying
the operator, fault=high location=throughput is reasoned, for it does not contradicts to the causal relation from throughput to inflow F1, outflow F2, valve V1, and level L1.

Typical single and multiple case operations are enumerated with the example of how it is selected.

• **Abstraction**: If the *effect type* distance of the related case from the current case is under a threshold and the related case is the supercase of the current case, then propose a case operation which guess the fault type of the current type by abstracting the fault type of the related case. e.g.

Given case:

```
(event name=target status=current type=data
       V1=open F2=low)
```

Related case:

```
(event name=Leak_41 type=data fault=leak
       V1=open L1=high F2=low)
```

the inferred result:

```
fault=FlowLost
```

• **Magnitude variation**: If the related case is interpreted as the same event except the magnitude, then make the magnitude of the given case in that way keeping event type same. This relation is recognized by matching only important attribute for that event type

(netflow DF, and valve V1 in this case), leaving away slight difference of other attributes. e.g.

Given case:

```
(event name=target status=current type=data
       F2=high V1=close DF=-)
```

Related case:

```
(event name=Leak_43
       type=data fault=leak
       F2=normal V1=close DF=-)
```

the inferred result:

```
fault=leak  and magnitude=big
```

Other than these operators for a single case, operators are prepared for multiple cases. If these nearest cases satisfy a certain relation with each other, case operations for multiple cases are applicable. The followings are typical examples of multiple case operators.

• **Extrapolation (or interpolation)**: If the related cases have values whose value assignment is corelated (F2 in this case) with fault type, then infer the unknown fault type by extrapolating these event types in the event space. e.g.

Given case:

```
(event name=target status=current type=data
       location=F2&V1 F2=low V1=normal)
```

Related cases:

```
(event name=LeakInF2&V1_2 type=data
       fault=leak location=F2&V1
       F2=high V1=normal)
(event name=NormalInF2&V1_1 type=data
       fault=normal location=F2&V1
       F2=normal V1=normal)
```

the inferred result:

```
fault=block location=F2&V1
```

• **Combination**: This case operator is used when there are two cases which are related as to *effect type*, but faults of these two cases seems to be independent. This operation combine the independent two faults.

As a typical multiple case operator, OPS83 rule used in CAOS is presented in Fig. 7.

```
rule FaultType_propose_combination_mul
{
&g   (goal PhaseStatus=current; step=propose;
           ProblemSpace=FaultType);
&e1  (event status=current);
&d   (distance from=&e1.id; type=effect;
           value<=&g.Edistance);
&d1  (distance from=&e1.id; to<>&d.to;
           type=effect;
           value<=&g.Edistance);
    -(distance from=&d.to; to=&d1.to;
           type=effect;
           value<0.5);
&e2  (event id=&d.to; type=data);
&e3  (event id=&d1.to; type=data);
    -(relation arg1=&e2.fault; arg2=&e3.fault;
           name=opposite);
    -(operator object=&d.to; object2=&d1.to;
      (@.PhaseStatus=done\/
       @.PhaseStatus=pending);
        ProblemSpace=combination );
-->
    modify &g (Edistance=0.0; Ddistance=0.0);
    make (operator id=gint ();
                PhaseStatus=pending ;
                status=proposed;
                ProblemSpace=combination;
                priority=5;
                object=&d.to;
                object2=&d1.to);
};
```

Fig. 7 OPS83 rule for case operation of combining independent two cases

English translation of this rule is:

> If there are two cases whose *effect type* distance to the current case is under the threshold, and there is no relation between these two cases as to *effect type*, then propose the candidate operator to guess the fault of the current case by combining the faults of these two cases.

In the rule, independency of these two related cases is expressed by the non-existency of relation case between them. The example showing how this operation behaves as follows:

Given case:

```
(event name=target status=current type=data
       F2=low L1=low V1=open)
```

Related cases:

```
(event name=BLockInF2&V1_15 type=data
       fault=block location=F2&V1
```

```
      F2=low V1=open)
(event name=AnormalyInControl_31 type=data
      fault=anormaly  location=control
      L1=low V1=open)
```

the inferred result:

```
fault=block location=F2&V1 and
fault=anormaly location=control
```

Abstraction (synthesis) is also possible for multiple case operator which abstract from leak, block to flow lost in terms of fault type, from high, low to abnormal as for flow, and so on. Case operations are summarized in Table 2.

4 Implementation and Sample Session

The experimental system is implemented for the process diagnosis example. It is implemented on the production system OPS83, where cases are structured at working memory as working memory elements with network structure. Case operation subsystem is written by about 100 rules including control mechanism of rules. The control mechanism uses automatic subgoaling mechanism of SEAR. About 20 cases are stored in the dynamic memory.

A sample session of case-based diagnosis by CAOS. The reasoning is carried out in the following order: (1) input of the current case, (2) reiminding of the nearest cases in terms of *data type* and *effect type*, (3) operate adequate case operation, and (4) output of the operated results. [2]

In this example, for the current case

```
(event  name=target status=current type=data
        V1=close F2=high L1=high)
```

where the fault is unknown, two cases are reminded, i.e.

```
(event name=LeakInF2&V1_7 type=data
      fault=leak location=F2&V1
      V1=close F2=high)
(event name=Anormaly_30 type=data
      fault=anormaly location=control
      V1=close L1=high)
```

by the reason that these two cases are nearest to the current case in terms of *effect type* distance. Since these two cases are considered to be independent of each other (confirmed by the fact that *effect types* of these two cases are not near), the case operator for combining these two cases is operated. As a result, the current case is diagnosed to have the multiple faults.

Table 2 Summary of Case Operations

case operator	Conditions the base case(s) must satisfy	Operation on related case(s)
SINGLE		
Abstraction	sub pattern(effect type)	abstract the fault type
Opposite Type	distance(effect type) <= threshold no relation(data type)	make the fault type opposite
Magnitude Variation	distance(effect type) <= threshold distance(data type) <= threshold sub pattern(data type) super pattern(data type)	 make the magnitude of fault bigger make the magnitude of fault smaller
MULTIPLE		
Extrapolation (Interpolation)	distance(data type) <= threshold One sympton(data type) in two related cases and given case change the same direction as that of values of faults in two related cases.	extrapolate (interpolate) the value of two faults
Combination	distances(effect type) <= threshold no relation between two related cases	combine two faults in conjunctive
Abstraction	distances(effect type) <= threshold related cases has opposite fault type	abstract the fault type

[2] These four tasks can be selected independently from the top menu.

```
**************************
  CAse  Operation  System
**************************

  1 Input new case
  2 Structure cases
  3 Case-based reasoning
  4 List structured cases
  5 Output case

Enter the number: 3

*********      Input the new case.

Input the case name: target

Input the attribute value pairs (E to end):
V1  close
F2  high
L1  high
E

*********     Reminding by data type.

target =super=> LeakInF2&V1_7
F2 high
V1 close

target =super=> Anormaly_30
V1 close
L1 high


*********    Reminding by effect type.

target =super=> Anormaly_30
L1 negative_V1
V1 negative_F2

target =super=> LeakInF2&V1_7
V1 negative_F2


*********    Searching for case operation.

data distance = 0.9
effect distance =0.9
operator combination is selected for cases:

LeakInF2&V1_7   Anormaly_30

*********    Output case operation results.


anormaly at Control
leak at F2&V1
```

Fig. 8 Sample Session of CAOS
(Inputs from user are underlined.)

Other than successful results, some failed reasoning done by CAOS will show the current status of the implementation. For the given case:

```
(event status=current V1=open F2=high
       type=data)
```

where a symptom F1=high is missed. Then the follwoing case is selected as the nearest in terms of *effect type* distance.

```
(event name=LeakInTank_9 V1=close F2=low
       fault=leak location=tank type=data)
```

Since the syndrome is opposite pattern of the current syndrome, the case operator which make the fault type opposite is selected. Thus, "tank is block" is concluded.If the data type of the current case is given properly, the case:

```
(event name=ThroughputLow_27 F1=low V1=close
       F2=low fault=low location=throughput
       type=data)
```

will be selected to operate this case operation. Then, correct diagnosis, "throughput is high", will be obtained.

Most of wrong diagnosis occurs reminding inappropriate cases for operation. This happens when qualitatively different two cases are placed near to each other in the structure. Other indexing besides *data type* and *effect type* must be introduced to avoid such wrong diagnosis, or rules must be modified to clearly separate case operations for superficially similar cases.

5 Comparison with the Rule-based Reasoning

There is no need for abstracting rules from cases in case-based reasoning, for cases themselves can be knowledge source. The generalization mechanism is built in the dynamic case-base. However, cases must be expressed in

such a manner that association of them is possible. Further, sample cases must be chosen from various types of cases, since the case-base of almost similar cases cannot respond flexibly to the many cases.

Although case-based and rule-based reasoning have fairly different knowledge representation and hence different reasoning mechanism, they are compatible in one system. There are many ways to combine these two paradigms. As an example, SEEK [Ginsberg et al 1988] is basically a rule-based system, but uses cases extensively to refine rules. CAOS uses rules as case transformation operators.

The combined paradigm of the case-based and the rule-based reasoning may contribute to break through the bottleneck of knowledge acquisition in most of knowledge-based systems. Rule-based system depends upon heuristic rules which are fixed once the domain knowledge are acquired, and does not have any dynamic part. This makes rule-base updated and maintained whenever the new rules are found. Knowledge acquisition problem arises from pushing permanent knowledge and variable knowledge into a uniform expression of rules. Assigning permanent knowledge to rule-base, and dynamic knowledge to case-base may lessen the necessity of modifying rule-base. However, as demonstrated in this paper, we must admit that the introduction of case-base causes the different problem of case acquisition summarized as: (1) Not only the amount but the variety of cases provided must be rich enough. (2) The order of cases input affects the structure of case-base.

6 Concluding Remarks

The architecture of case-based diagnosis presented shows that diagnosis by the case-base indexed by causal relation among symptoms becomes more flexible than diagnosis by that indexed by symptoms. This resembles the fact thata case can remind the human experts of the case with deeper similarities, while the same case remind the novices of the cases with superficial similarity. The introduction of the indexing by a new aspect will lead to more interesting reminding, and hence reasoning.

Although CAOS is still in the development stage especially in case operation subsystem, the only reminding subsystem without case operating system seems to be useful as an intelligent decision-support for. human experts who need previous similar cases to analyze the current problem system.

Implications as to the case acquisition strategy such as; cases should be sampled from a widespread variety of types, and the order of cases input affects the structure of the case-base, are obtained from the experiments of CAOS.

References

[Schank 1986] Roger C. Schank. *Explanation Patterns.* LEA Publishers, London, 1986.

[Winston 1982] Patric H. Winston. Learning Principles from Precedents and Exercises. *Artificial Intelligence*,19(3), 1982.

[Carbonell 1985] Jaime Carbonell. Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. Carnegie-Mellon University, cmu-cs-85-115, 1985.

[Schank 1982] Roger C. Schank. *Dynamic Memory.* Cambridge University Press, 1982.

[Lebowitz 1982] Michael Lebowitz. Correcting Erroneous Generalizations, *Cognition and Brain Theory 5*, :367-381 1982.

[Kolondner 1986] Janet L. Kolondner. A Process Model of Case-based Reasoning in problem Solving, *Proc. of AAAI-86* :284-290, 1986.

[Bain 1986] William M. Bain, A Case-based Reasoning System in Subjective Assessment, *Proc. of AAAI-86* :523-527, 1986.

[Hammond 1986] Kristian J. Hammond. SHEF: A Model of Case-bsed Planning, *Proc. of AAAI-86* :267-271, 1986.

[Belenblut and Whitehouse 1973] B.J. Belenblut and M. B. Whitehouse. A Method for Monitoring Process Plant Based on Decision Table Analysis. *chem. Eng.*, :175-181, 1973.

[Forgy 1986] Chales L. Forgy. OPS 83 User's Manual. Carnegie-Mellon University, 1986.

[van de Brug 1986] A. van de Brug, J. Bachant, and J. McDermott. Taming of R1. *IEEE Expert*, 1(3):33-38, 1986.

[Ginsberg et al 1988] Allan Ginsberg, Solomon M. Weiss and Peter Politakis. Automatic Knowledge Base Refinement for Classification System. *Artificial Intelligence*, 35:197-226, 1988.

Parallel Solutions to Constraint Satisfaction Problems

extended abstract

Simon Kasif
Department of Computer Science
The Johns Hopkins University
Baltimore, Md 21218

1. Introduction

One of the key problems facing Artificial Intelligence (AI) is performing efficient inferences over large knowledge bases. Viewing this problem in the context of parallel computation poses fundamental research problems such knowledge representation in a parallel environment, problem decomposition methods, classes of (AI) problems which are amenable to parallel computation, the role of incremental methods, parallel matching, and parallel search strategies that eliminate redundancy.

It is generally believed that humans perform many tasks efficiently (i.e., in almost constant time) due to the exploitation of massive parallelism in the brain (e.g., Feldman's 100 instruction step metaphor). In a view of parallelism advocated by many AI researchers (e.g., Connection Machine) we associate a small simple processor with each data element (thus loading data does not require time). In this perspective logarithmic time parallel algorithms have important implications since they show how to represent the data and perform parallel search in almost constant time for any reasonable size network. In this paper, we relate parallel constraint networks to standard models of computation (PRAMS). We present several basic techniques for achieving parallel execution of constraint networks. We are primarily interested in developing a classification of constraint networks that admit massively parallel execution. We show that contrary to the common intuition, chain networks do admit parallel solutions (as in fact do all acyclic graphs). We also present a parallel solution for general constraint graphs with the restriction that only two labels are allowed per variable. The major result supported by our recent investigations reported in this paper and in [Kasif & Delcher 88] is that the parallel complexity of constraint networks is critically dependent on subtle properties of the network that do not influence its sequential complexity.

2. Constraint Satisfaction Networks

Constraint satisfaction networks have been shown to be an important tool for modelling a variety of Artificial Intelligence applications [Winston 84], [de Kleer 86]. These networks often utilize local constraint propagation techniques to achieve global consistency (consistent labelling in vision). Such methods have been used extensively [Rosenfeld et al. 76], [Haralick & Shapiro 79], [Mackworth 77] as well as planning, natural language analysis and common-sense reasoning (truth maintenance systems) [Winston 84], [de Kleer 86].

A commonly accepted formalization of constraint networks is in terms of *a constraint satisfaction problem (CSP),* sometimes referred to as the consistent labelling problem (CLP).

The constraint satisfaction problem can be informally defined as follows. Let S be a set of objects. Each object has a set of possible labels associated with it. Additionally, we are given a set of constraints that for each object s and label x describe the compatibility of assigning the label x to object s with assignment of any other label x' to any other object s' .

An interesting approach to model CSP problems is by means of a constraint graph. The nodes of the constraint graph correspond to variables of CSP. The edges of the graph correspond to the binary constraints in the CSP. That is, with each edge in the constraint graph we associate a matrix that shows which assignments of labels to variables on the arc are permitted. In this interpretation CSP can be seen as generalized graph coloring

Since CSP is known to be NP-complete, the discrete relaxation method has been proposed to reduce the initial ambiguity. Arc Consistency (AC) allows an assignment of a label x to an object s iff for any other object s' in the domain there exists a valid assignment of a label x' which does not violate the constraints (a formal definition is given in the next section). This formalization allows us to achieve global consistency in many cases by local propagation of constraints. Specifically, a discrete relaxation algorithm can discard a label from an object if it is incompatible with all other possible assignments of labels to the remaining objects. The discrete relaxation approach has been successfully applied to numerous computer vision applications [Waltz 75.], [Kitchen 1980], [Barrow & Tenenbaum 76], [Brooks 81]. The sequential time complexity of AC is discussed in [Mackworth & Freuder 85].

Since the constraint propagation procedures such as discrete relaxation appear to operate locally, it has been previously believed that the relaxation approach for CSP has a natural parallel solution [Rosenfeld et al. 76], [Ballard & Brown 82], [Winston 84]. Our analysis, presented in [Kasif 86] suggests that a parallel solution is unlikely to improve by much the known sequential solutions. Specifically, we proved that arc consistency belongs to the class of inherently sequential problems called log-space complete for P (or P-complete).

Intuitively, a problem is P-complete iff a logarithmic time parallel solution (with polynomial number of processors) for the problem will produce a logarithmic time parallel solution for any polynomial time deterministic sequential algorithm. This implies that unless P $\subseteq$ NC (the class of problems solvable in logarithmic parallel time with polynomial number of processors) we cannot solve the problem in logarithmic time using a polynomial number of processors.

In the next sections we give a formal definition of a constraint satisfaction problem and provide several parallel techniques.

3. Constraint Satisfaction and Discrete Relaxation

The constraint satisfaction problem (CSP) and its less restrictive versions are formally defined in [Mackworth 77] and [Rosenfeld et al. 76]. For completeness we give a semiformal definition here. Let $V=\{ v_1, \ldots, v_n \}$ be a set of variables. With each variable v_i we associate a set of labels L_i. Now let P_{ij} be a binary predicate that defines the compatibility of assigning labels to objects. Specifically,

$$P_{ij}(x, y)=1$$

iff the assignment of label x to v_i is compatible with the assignment of label y to v_j. The Constraint Satisfaction Problem (CSP) is defined as the problem of finding an assignment of labels to the variables that does not violate the constraints given by P_{ij}. More formally, a solution to CSP is a vector $(x_1, \ldots, x_n)$ such that x_i is in L_i and for each i and j, $P_{ij}(x_i, x_j) = 1$.

For example, the 4-queens problem can be seen as an instance of CSP. To confirm this, associate a variable with each column in the

board and let $L_i = \{1,2,3,4\}$ for $1 \leq i \leq 4$. Let $P_{ij}(x,y)=1$ iff positioning of a queen in row x at column i is " safe " when there is a queen in column j and row y. As mentioned in the introduction, the CSP is known to be NP-complete. Therefore several polynomial approximation algorithms were proposed and were shown to perform quite well in practical applications. The most significant class of algorithms are variations on discrete relaxation [Rosenfeld et al. 76] also known as local network consistency algorithms [Mackworth 77]. Formally, a solution to the local version of CSP (arc consistency) is a set of sets $M_1, \ldots, M_n$ such that M_i is a subset of L_i and a label x is in M_i iff for every M_j $i \neq j$ there is a y_{xj} in M_j, such that $P_{ij}(x,y_{xj}) = 1$. Intuitively, a label x is assigned to a variable iff for every other variable there is at least one valid assignment of a label to that other variable that supports the assignment of label x to the first variable. This condition is most commonly referred to as arc consistency (AC) [Mackworth 77]. That is, for every two nodes in the constraint graph we verify that the current set of possible assignments to nodes residing on the arc are consistent in the sense of being a potential solution. Clearly, a solution to arc consistency is a necessary condition to any solution to CSP but it is not a sufficient condition. We call a set $M_1, \ldots, M_n$ to be a maximal solution for AC iff there does not exist any other solution $S_1, \ldots, S_n$ such that $M_i \subseteq S_i$ for all $1 \leq i \leq n$. We are only interested in maximal solutions for an AC problem. This restriction is necessary since any solution for a AC represents a set of candidate solutions for the original CSP, which will eventually be verified by a final exhaustive check. Thus, by insisting on maximality we guarantee that we are not losing any possible solutions for the original CSP. Therefore, in the remainder of this paper a solution for an AC problem is identified with a maximal solution.

4. Parallel Processing of Constraint Networks

The standard approach for achieving arc consistency is repeatedly applying the following procedure (discrete relaxation)

Par-AC

for each directed arc from X_i to X_j of the constraint graph test whether
for each label l for X_j there exist l' for X_i that permits it.

It is easy to see that this algorithm will terminate in $O(EK^2nK)$ time, where E is the number of edges in the graph and K is the number of labels, (recall that EK^2 is the size of the input). In fact, [Mackworth & Freuder 85] and this paper develop much better sequential algorithms for the problem. Clearly procedure **Par-AC** can be parallelized in a straightforward way. If we assume CRCW PRAM as our model of parallel computation we have the following simple result: CRCW PRAM is a standard shared memory parallel computation model that permits concurrent read and writes into the same location. Concurrent writes are permitted if they agree on the written data.

Claim 1: The parallel complexity of procedure **Par-AC** is $O(nK)$ on a CRCW PRAM.

Proof:

Simply attach a processor to each arc and label and repeatedly test the arc consistency condition. Arc consistency is essentially an OR on set membership of a set of labels which can be performed in constant time on CRCW PRAM. At each parallel step a label must be dropped and there are at most nK labels. □

On a more realistic model of computation such as EREW (exclusive read/write) PRAM we can perform the above procedure in $O(nK \log K)$ parallel time.

The usual way to see parallel computation of constraint networks in the AI community is

not via shared memory models such as the CRCW PRAM. We usually assume that we have processors associated with nodes/arcs of the network. The processors communicate to their neighbors. It is obvious that this perspective requires the full power of a CRCW PRAM model for the following reasons.

1. We assume constant overhead for communication between adjacent processors, i.e., complete graph connectivity.
2. We assume at any time all the neighbors of a node can communicate with the node (i.e., propagate constraints, remove labels, etc), that is concurrent read/write capabilities.

It is easy to see that the procedure above has a lower bound of nK steps, i.e., does not parallelize in the worst case. As a simple example, consider a chain symmetric constraint graph X1-X2-X3-,...,-Xn with label set {0,1} and assume the only supporting assignment for a 0 is 0. All the initial label sets for the variables are {0,1} except S1 = {1}. In [Kasif 86] we proved a much stronger result namely that no procedure is likely to parallelize in the worst case.

Theorem 2: (Kasif 86)

The Propositional Horn clause satisfiability problem is log-space reducible to the AC problem. That is, AC is P-complete.

Intuitively, this result states that to achieve logarithmic parallel time one would (probably) need an exponential number of processors (in the worst case). This is indeed a worst case result and indeed [Dixon & de Kleer 88] reported a successful experiment with massively parallel constraint processing.

One intuitive but incorrect interpretation of the Theorem 2 is that inherent sequentiality is caused because of long constraint chains such as the one below.

X1---X2---X3--------...---Xn

We give a simple technique for parallel processing of constraint chains or more generally trees. Most importantly we show that while the degree of parallelism is dependent on the diameter of the constraint graph, a long diameter is not a sufficient condition for inherent sequentiality of constraint networks.

We first make an important observation which is critical to the understanding of the complexity of achieving arc consistency. Given a constraint satisfaction problem S one can construct a propositional Horn formula (AND/OR graph) G such that arc consistency of S can be achieved by (essentially) running a satisfiability algorithm for G. For a formal construction see [Kasif 85]. We sketch the intuition here. For each label l and a variable X we construct a propositional atom $P_{X,l}$ which means l stays at X. Consider a variable X connected in the constraint graph to variables Y and Z. Assume that Y has labels $l1$, $l2$ and Z has labels $l3$ $l4$ that support l at X. Thus, we construct the formulae:

$$P_{Y,l1} \rightarrow P12$$
$$P_{Y,l2} \rightarrow P12$$
$$P_{Z,l3} \rightarrow P34$$
$$P_{Z,l4} \rightarrow P34$$
$$P12, P34 \rightarrow P_{X,l}$$

Note that if the constraint graph has E edges and K labels per variable, the size of the formulae is EK^2. More importantly, the sequential complexity of solving satisfiability of this graph is $O(EK^2)$ (see [Kasif 85] for details. This is a slight improvement over the result in [Mackworth & Freuder 85]. Note, that this reduction is from AC to propositional Horn solvability (PHS) as opposed to the one used to prove Theorem 2 (from PHS to AC).

We first show that if the number of possible labels per variable is two or less we can process the constraint graph in logarithmic parallel time.

Lemma 3:

Given a CSP P, where the $|S_i| \leq 2$, that is number of possible labels (solutions) per variable is at most 2, then P can be converted to a 2-SAT problem (satisfiability of clauses with two literals per clause) S such that I is a solution for S iff I is a solution for P.

Proof: (sketch)

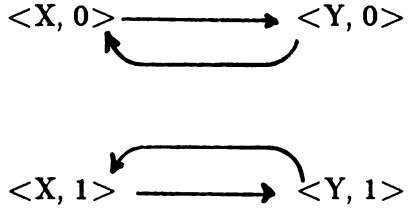
We convert each one of the arcs to a 2-CNF formula (formula in conjunctive normal form with at most two variables in any disjunction). Details are omitted. For instance, the constraint

X	Y	$P_{X,Y}$
0	0	0
0	1	1
1	0	1
1	1	0

is converted to the set of clauses

$$c(\neg X \vee Y),(X \vee \neg Y). \square$$

Given a set of clauses in 2-SAT form, we create a new directed constraint graph that captures the constraints of the initial constraint graph more explicitly. For each disjunction of the form A V B we create a set of 4 nodes <A,0>,<A,1>,<B,0> and <B,1>. <A,L> is connected with a directed arc to <Y,L' > iff assigning L to X forces assigning L' to Y in order to evaluate the disjunction to true. For the constraint above expressed by the formula (¬ X V Y),(X V ¬ Y) we generate the following graph.



Using the construction of an explicit directed graph we can find solutions to the original problem by tracing the strongly connected components of the constraint graph. This method is a systematic application of deterministic constraint pushing employed in several constraint systems. The precise claim is captured in the next intuitive proposition stated without proof. The formal proof is involved.

Proposition 4:

Let S be a 2-SAT formula and C be its explicit constraint graph constructed as above. S is satisfiable iff $X=0$ and $X=1$ do not belong to the same strongly connected component of C for any variable X in S. Moreover, if S is satisfiable, the directed components of C can be used to derive an assignments for the variables of S.

Computing directed components in parallel is a well understood problem and is done by transitive closure methods.

Proposition 5:

Given a CSP S where the $|S_i| \leq 2$, that is, the number of possible labels (solutions) per variable is at most 2, S has a poly-logarithmic parallel time solution $O(\log^2 N)$ with N^3 processors.

Thus, in the special case when the number of labels per object is at most than 2, we can compute the solution in sublinear time with a relatively large number of processors ($N^{1.5}$ processors where N is the size of the input). We note that we are actually solving the constraint problem directly without attempting to achieve arc consistency first.

The next set of results utilizes the topology of the constraint graph to achieve parallelism. We take advantage of the separability properties of the constraint network for recursive parallel decomposition of the problem. A more powerful (but higher overhead method was presented for planar constraint networks in [Kasif et al 1987] (see also [Seidel 81]).

Lemma 6:

Given any constraint tree T of size |T| = N, we can construct a binary constraint tree T' with a constant increase in size and T and T' have the same solutions.

Proof

The proof is trivial and is omitted. We perform the standard n-ary to binary tree transformation and copy the appropriate constraints to the new arcs. □

Notice we are not transforming n-ary constraints networks into binary constraint networks; we just modify the topology of the original constraint tree with a constant increase in size. Thus, without loss of generality we consider binary trees only.

Lemma 7: Given any tree T of size |T| = N, there is a node n_0 in T such that the subtree T1 rooted at n_0 that satisfies 1/3N < |T1| < 2/3N.

Proof (sketch)

Start from the root, if one of the subtrees satisfies this condition we, are done. Otherwise, follow the path in the direction of the larger tree and repeat. □

Proposition 8: If the constraint graph is a connected acyclic graph (i.e. tree) we can solve the CSP for the graphs in $O(logN)$ time with $O(E)$ processors.

Proof (sketch)

Find a node n_0 in T and a subtree T1 rooted under n_0 such that 1/3N < |T1| < 2/3N. Disconnect n_0 and its adjacent edges, thus generating two disconnected trees. Recursively solve CSP for the two trees. The recurrence is

T(N) ≤ C*Max{TN(2/3N),TN(1/3N)}+C1,

where C, and C1 are constants. Thus, we can solve the CSP in poly-logarithmic time with a polynomial number of processors (that depends on K, the number of labels per variable). □

A similar method is possible for solving the AC problem and is discussed in the full version of the paper.

5. Discussion

In this paper we presented several techniques for parallel processing of constraint networks. Our methods are motivated by graph theoretical considerations, similar to the methodology of [Dechter & Pearl 88]. Surprisingly, the separator based methods fail even for chain graphs when the number of labels per variable is large (proportional to the size of the graph) It turns out that in this case the parallel complexity of processing constraint graphs is critically dependent on the specifics of the constraint predicates. In [Kasif & Delcher 88] we show that consideration such as whether the constraint graph is directed, symmetric and the number of labels per variable play an important role in the parallel solution.

We mentioned before the direct connection between AND/OR graph solvability and constraint satisfaction. Parallel AND/OR problem reduction is another important AI technique. based on the concepts of decomposability in production systems. Consider the simple partial program below:

```
SOLVE(P) :- SOLVE(P1),SOLVE(P2).
SOLVE(P) :- SOLVE(P3),SOLVE(P4).
...
```

This program asserts that in order to solve problem P we can either solve problems P1 and P2 or solve problems P3 and P4. AND-parallelism refers to the parallel solution of conjunctive goals such as P1 and P2. This kind of parallelism is found in conventional languages where we execute several program statements in parallel. OR-parallelism addresses the parallel activation of both clauses that may solve the problem P. OR-parallelism is typical in AI problems where there may be many rules initially applicable to a given

situation.

The experimental evidence accumulated by many researchers thus far seems to suggest that only limited parallelism is attainable in large scale knowledge base (production) systems [Symp. 88]. This could be an inherent feature of a particular problem domain or the result of lack of adequate tools for exploring the sources for parallelism. We believe that although achieving parallelism is a non-trivial complex task, it is nevertheless possible by choosing the right representation of data and proper problem decomposition. Consider the following simple example:

$$r(1) \leftarrow r(2).$$
$$r(2) \leftarrow r(3).$$
$$\cdots$$
$$\cdots$$
$$r(n\text{-}1) \leftarrow r(n).$$

On the surface, this appears to be as an inherently sequential set of logical implications, and indeed it does not contain any AND/OR parallelism. However, using the tree contraction technique developed by Miller & Reif we can compute perform this sequence of rule application in parallel. The basic idea is as follows. We use the following three transformation rules.

1. If A $\leftarrow$ B is true and B $\leftarrow$ C are true then A $\leftarrow$ C is true.
2. If A $\leftarrow$ B,C; and C is true, we create the fact A $\leftarrow$ B.
3. If A $\leftarrow$ B,C is true and B and C are true so is A.

It can be shown that for many inference networks the above procedure generates optimal parallel execution. For example, in the case of the program above, we repeatedly use rule 1 which corresponds to "parallel doubling" and we get the following execution.

1. Parallel Step 1:
   $r(0) \leftarrow r(2)$; $r(2) \leftarrow r(4)$; $r(4) \leftarrow r(6)$;...
2. Parallel Step 2:
   $r(0) \leftarrow r(4)$; $r(4) \leftarrow r(8)$; $r(8) \leftarrow r(12)$;...
   Parallel Step 3:

$r(0) \leftarrow r(8)$; $r(8) \leftarrow r(16)$;..

...

The main idea is dynamic tree restructuring (proof tree) that allows full utilization of the available processors.

The example above illustrates that it is sometimes possible to generate a parallel execution in seemingly sequential problems by employing slightly more sophisticated techniques. This paper made a step in that direction in the context of constraint networks.

Acknowledgements

This research has been partially supported by the Air Force Office of Scientific Research under grant AFOSR-89-1151 and National Science Foundation under grant IRI-88-09324. Thanks are due to Johan de Kleer, John Reif, and Dave Waltz for their constructive comments that motivated this research.

REFERENCES

[Symp. 88]
Stanford Symposium on Parallel Models of Intelligence. 1988.

[Ballard & Brown 82]
Ballard, D.H. and C.M. Brown, *Computer Vision,* Prentice Hall, 1982.

[Barrow & Tenenbaum 76]
Barrow, H. G. and J. M. Tenenbaum, MSYS: A system for reasoning about scenes, Technical Note 121, SRI AI Center, Menlo Park, CA, April 1976.

[Brooks 81]
Brooks, R A., Symbolic reasoning among 3-D models and 2-D images, *Artificial Intelligence* **17**, pp. 285-348, 1981.

[Dechter & Pearl 88]
Dechter, R. and J. Pearl, Tree Clustering Schemes for Constraint Processing, *Proceedings of AAAI-88*, pp. 150-154, 1988.

[Dixon & de Kleer 88]
Dixon, M. and J. de Kleer, Massively Parallel Assumption-based Truth Maintenance, *Proceedings of AAAI-88*, pp. 199-204, 1988.

[Haralick & Shapiro 79]
Haralick, R. M. and L. G. Shapiro, The consistent labeling problem: Part I, *IEEE Trans. Patt. Anal. Mach. Intel.* **PAMI-1**, pp. 173-184, 1979.

[Kasif 85]
Kasif, S., On the Parallel Complexity of Discrete Relaxation, JHU/EECS-85/18 (accepted for publication AI Journal), 1985.

[Kasif 86]
Kasif, S., On the Parallel Complexity of Some Constraint Satisfaction Problems, *Proceedings of the 1986 National Conference on Artificial Intelligence*, August 1986.

[Kasif et al 1987]
Kasif, S., J. Reif, and D. Sherlekar, Formula Dissection: A Parallel Algorithm for Constraint Satisfaction, *Proceedings of the 1987 IEEE Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence*, October 1987.

[Kasif & Delcher 88]
Kasif, S. and A. Delcher, Analysis of Parallelism in Constraint Satisfaction Networks, JHU/CS-88/30, 1988.

[Kitchen 1980]
Kitchen, L. J., Relaxation applied to matching quantitative relational structures, *IEEE Trans. Syst. Man Cybern.* **SMC-10**, pp. 96-101, 1980.

[de Kleer 86]
Kleer, J. de, An Assumption based TMS, *Artificial Intelligence* **28**, pp. 127-162, 1986.

[Mackworth & Freuder 85]
Mackworth, A. and E. Freuder, The complexity of some polynomial network consistency algorithms for constraint satisfaction, *Artificial Intelligence* **25**, pp. 65-74, 1985.

[Mackworth 77]
Mackworth, A. K., Consistency in networks of relations, *Artificial Intelligence* **8**, pp. 99-118, 1977.

[Rosenfeld et al. 76]
Rosenfeld, A., R. Hummel, and S. Zucker,, Scene labeling by relaxation operations, *IEEE Trans. Syst. Man Cybern* **SMC-6**, pp. 420-433, 1976.

[Seidel 81]
Seidel, R., A New Method For Solving Constraint Satisfaction Problems, *Proceedings of the International Joint Conference on AI*, pp. 338-342, 1981.

[Waltz 75.]
Waltz, D., Understanding line drawings of scenes with shadows, pp. 19-92 in *The Psychology of Computer Vision,* ed. P. H. Winston, McGraw-Hill, New York, 1975..

[Winston 84]
Winston, P.H., *Artificial Intelligence,* Addison Wesley, 1984.

Hard Problems for Simple Default Logics

Henry A. Kautz
AT&T Bell Laboratories
Murray Hill, N.J. 01774
kautz@allegra.att.com

Bart Selman
Dept. of Computer Science
University of Toronto
Toronto, Canada M5S 1A4
bart@ai.toronto.edu

Abstract

We investigate the complexity of reasoning with a number of limited default logics. Surprising negative results (the high complexity of simple three literal default rules) as well as positive results (a fast algorithm for skeptical reasoning with binary defaults) are reported, and sources of complexity are discussed. These results impact on work on defeasible inheritance hierarchies as well as default reasoning in general.

1 Introduction

It has been suggested that some kind of default inference can be used to simplify and speed common sense reasoning. Researchers have appealed to default logics as a solution to the problem of generating and reasoning with large numbers of "frame axioms"; as a way of simplifying complex probabilistic calculations; and recently as a way of "vivifying" (filling out) an incomplete knowledge base, thus suppressing the complexities of reasoning with uncertainty [McCarthy, 1985, Levesque, 1986].

While current formal theories of default inference are computationally much worse than ordinary logic, it has been tacitly assumed that this additional complexity arises from their use of consistency tests. Our interest in fast, special purpose inference mechanisms led us to investigate very simple propositional, disjunction-free systems of default reasoning, where consistency checking is trivial. Here, we thought, default reasoning should shine.

This paper reports a number of surprising complexity results involving restricted versions of Ray Reiter's Default Logic [Reiter, 1980]. We define a partially-ordered space of more and less general propositional default theories. For each we determine the complexity of solving the following three problems: finding an extension; determining if a given proposition is true in some extension; and determining if a given proposition is true in all extensions. While all of these problems are NP-hard (or co-NP-hard) for unary theories (which roughly correspond to inheritance graphs), they are all tractable for unary normal theories (which correspond to inheritance graphs without any specificity ordering). Other limited systems demonstrate that the three kinds of problems are strictly ordered in order of difficulty.

Finally we provide an intuitive characterization of sources of complexity in default reasoning.

Only a limited number of proofs are presented in this paper; all proofs and further discussion will appear in [Kautz and Selman, 1989].

2 Reiter's Default Logic

Reiter formalized default reasoning by extending first-order logic with default rules. This paper will not consider the other non-monotonic formalisms based on modal logic, circumscription, or model preference rules, although many of the results it presents have counterparts in those systems. (See [Selman and Kautz, 1988] for a similar analysis of model-preference theories.)

A *default theory* is a pair (D, W) where D is a set of default rules and W a set of ordinary first-order wffs. A rule is of the form:

$$\frac{\alpha : \beta \wedge \gamma}{\beta}$$

where α is the *prerequisite*, β the *conclusion*, and $\beta \wedge \gamma$ the *justification* of rule, each of them wffs. The rule is intuitively understood as meaning that if α is known, and $\beta \wedge \gamma$ is consistent with what is known, then β may be inferred. If γ is missing then the rule is *normal*; otherwise, it is *semi-normal*.

An *extension* is a maximal set of conclusions that can be drawn from a theory. But care must be taken that none of the justifications of the rules used in the construction of an extension conflict with its final contents, and that every wff in the extension can in fact be derived from W and the rules. The formal definition of an extension (from [Reiter, 1980]) is therefore rather complex. For any closed set of wffs S let $\Gamma(S)$ be the smallest set satisfying the following three properties:

1. $W \subseteq \Gamma(S)$
2. $\Gamma(S)$ is deductively closed.

3. If there is a rule such that $\alpha \in \Gamma(S)$, and $\neg(\beta \wedge \gamma) \notin S$, then $\beta \in \Gamma(S)$.

An extension is defined to be a fixed-point of Γ, so $E = \Gamma(E)$. A theory can have several, one, or no extensions.

Although normal theories have a number of nice theoretical and computational properties, semi-normal rules are often needed to establish a priority among the defaults. For examples, two default rules may have conflicting conclusions, yet have their preconditions satisfied in the same situation. If normal rules were used, this kind of situation would lead to two different extensions. One may know, however, that the first rule should always take priority over the second when both apply. This can be encoded by adding the negation of the precondition of the first rule to the justification of the second rule. Formally, given rules D_1 and D_2, where

$$D_1 = \frac{\alpha_1 : \beta_1 \wedge \gamma_1}{\beta_1} \qquad D_2 = \frac{\alpha_2 : \beta_2 \wedge \gamma_2}{\beta_2}$$

in order to establish D_1 as being of higher priority than D_2, replace D_2 by D_2':

$$D_2' = \frac{\alpha_2 : \beta_2 \wedge \gamma_2 \wedge \overline{\alpha_1}}{\beta_2}$$

One kind of priority that this scheme can encode is the "specificity" ordering that intuitively should appear in an inheritance hierarchy. For example, W may include the fact that "penguins are birds",[1] and D defaults that assert that penguins don't fly, and that birds do fly. The first, more specific default can be given priority over the second by encoding the pair as

$$\frac{Penguin : \overline{Fly}}{\overline{Fly}} \qquad \frac{Bird : Fly \wedge \overline{Penguin}}{Fly}$$

3 Complexity

Following [Garey and Johnson, 1979], we shall refer to a problem class as "tractable" if a polynomial-time algorithm can solve all its instances. While it is usually very difficult to to prove that a problem is intractable, it is often possible to show that it is as hard as any solvable in polynomial time by a *non-deterministic* computer. Such a reduction to an "NP-complete" problem indicates intractability for all practical purposes: no one seriously doubts that such problems require exponential time, even though it has not been so proven.

The NP-complete problem "3SAT" is that of determining the satisfiability of a conjunction of three-element clauses in propositional logic; that is, of a formula of the form:

$$\sigma = (x_1 \vee y_1 \vee z_1) \wedge (x_2 \vee y_2 \vee z_2) \wedge \cdots$$

[1] It remains an open problem to determine if a default theory *must* include this assertion, although a survey of the literature lends strong evidence to the conjecture. Certainly it is true that every *paper* on non-monotonic reasoning must include this example [Etherington *et al.*, 1989a].

For each default logic problem considered in this paper, we will present either a polynomial-time algorithm to solve its instances, or a reduction of the problem class to 3SAT. In the case of a reduction we'll call the problem NP-hard, without any implication that the problems are not in NP (and thus NP-*complete*). (In some cases we'll actually show that a problem is "co-NP-hard", meaning that it is as hard as any whose complement is in NP. For example, while determining that a formula is satisfiable is in NP, determining that a formula is not satisfiable is in co-NP. The distinction is not important for the purposes of this paper. Although it is unknown whether NP=co-NP, almost certainly neither are equal to P.)

Coarse as it may be, this division into tractable and intractable problems is an extremely useful one. We feel safe in dismissing claims that massively parallel computers can be used to "solve" NP-hard problems. (The chart on [Garey and Johnson, 1979, page 7] is instructive in this regard; a problem that would require 100,000 steps using an $O(n^3)$ algorithm would require over 10^{13} steps with an $O(2^n)$ algorithm.)

Furthermore, we find it difficult to appreciate work on parallel marker-passing algorithms for special cases of default reasoning until we know if the particular problems being solved have a practical serial solution, and can calculate the speedup the parallel solution yields.

A valid criticism of this analysis is that we should go on to develop fast algorithms which yield an approximate solution to the intractable problems. We do not yet have a clear idea of how to define a "approximate solution" to a problem in logical reasoning.

4 A Taxonomy of Default Theories

Two sources of complexity in default theories are readily apparent: the inherent complexity of the first-order component (W), and the complexity of determining whether the justification of a default rule is consistent with the currently-derived set of formulas. We'll restrict our attention to finite propositional theories in which W is simply a set (conjunction) of literals. The precondition, justification, and consequence of each default rule must be a conjunction of literals. Thus determining whether a default rule is applicable to W is trivial: the precondition must be a subset of W, and the intersection of W with the negation of each literal in the justification must be empty. The extended theory is again a set (conjunction) of literals. Although an extension is, by definition, an infinite, deductively closed set of formulas, any extension of a finite disjunction-free theory is logically equivalent to a finite set of literals. Henceforth when we speak of "computing an extension", we will mean computing such a finite set of literals.

Any inferential power such systems possess resides in the default rules; the only non-default inference rules

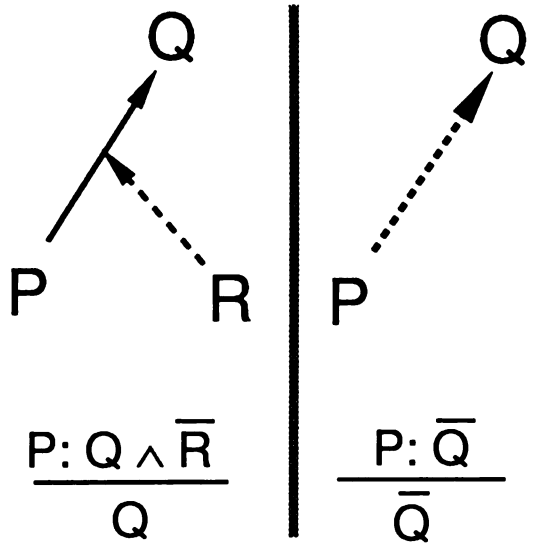


Figure 1: Unary Default Theories

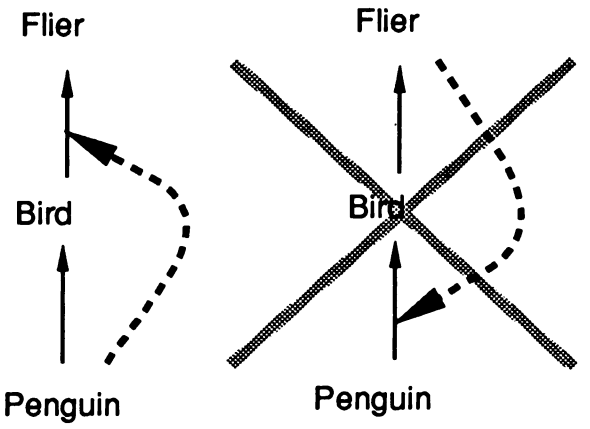


Figure 2: Cycles involving cancel arcs are prohibited in ordered theories.

which apply are conjunction-in and –out (to convert, e.g., $\{\alpha, \beta\}$ to $\alpha \wedge \beta$ and vice-versa). The reader should remember, in particular, that because the default rules are in fact rules and not axioms, the principle of reasoning by cases does not apply. For example, given a theory with empty W and rules

$$\frac{p : q}{q} \qquad \frac{\overline{p} : q}{q}$$

one may *not* conclude q.

Further restrictions on the form of the default rules leads to the following kinds of theories.

Unary These theories restrict the prerequisite to a single positive literal; the consequence to a single literal; and the justification to either the consequence or, in the case of a positive consequence, to the conjunction of the consequence and a single negative literal. These theories have a simple graphical notation, as shown in figure 1. Positive and negative default arcs appear, where optional cancel links may be attached to positive arcs. Unary theories are a special case of Etherington's "network theories" [Etherington, 1988, page 91], which also allow negative literals to appear in the prerequisite, and binary disjunctions to appear in W.

Disjunction-Free Ordered In an ordered theory the default rules induce a partial order on the set of literals. While the exact definition of an ordered theory is complex [Etherington, 1988, page 86], the intuitive goal is to insure that *some* sequence of default rule applications exists in which the application of a rule never undercuts the justification of a previously applied rule. In the disjunction-free case, a rule of the form:

$$\frac{a_1 \wedge \cdots \wedge a_{n_a} : b_1 \wedge \cdots \wedge b_{n_b} \wedge c_1 \wedge \cdots \wedge c_{n_c}}{b_1 \wedge \cdots \wedge b_{n_b}}$$

orders the literals as follows, for all i and j:

$$a_i \leq b_j \qquad \overline{c_i} < b_j \qquad \overline{b_i} \leq b_j$$

where $<$ and $\leq$ are transitive and related as usual. It must not be the case that $p < p$ for any literal in an ordered theory.

Ordered Unary These unary theories have no cycles involving cancel arcs, as shown in figure 2. Ordered unary theories appear to possess the minimum amount of machinery necessary to represent inheritance hierarchies with prioritized rules.

Disjunction-Free Normal The justification of each default rule in a normal theory is exactly the same as the conclusion.

Horn In these theories the literals in the prerequisite are all positive, and the justification and consequence is a single literal.

Normal Unary The graphical representation of these theories contain only positive and negative default implication arcs, with no cancellation arcs. There is no way for a more specific default to override a less specific one in these theories.

The sets of such theories form a partial order, as shown in figure 3. A negative complexity result (that is, a reduction to an NP-Hard problem) for an element in the partial order applies also to all elements above it. A positive complexity result (that is, a polynomial time algorithm) for an element applies also to all elements below it.

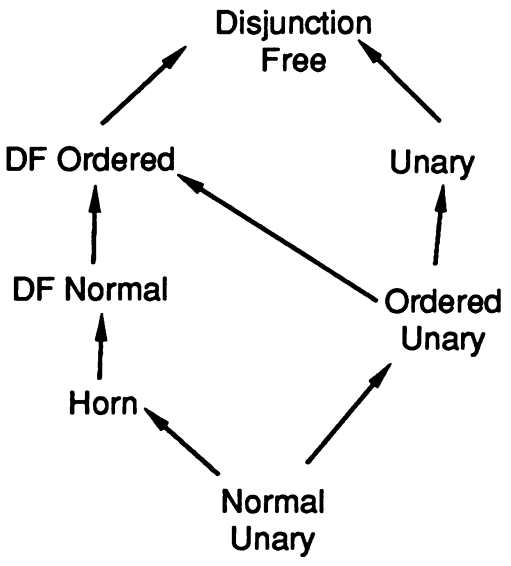


Figure 3: The hierarchy of default theories.

5 Finding an Extension

Extensions are defined by a fixpoint construction, and considerable effort has gone into the problem of converting that definition into an algorithm. Not all theories have extensions, and indeed, the very question of whether an arbitrary theory has an extension is undecidable. But a positive result exists for ordered theories:

Theorem 1 *There is an $O(n^2)$ algorithm which finds an extension of a finite, propositional, disjunction-free ordered theory, where n is the number of default rules in the theory.*

A sound and complete general non-deterministic algorithm for finding extensions of ordered theories appears in [Etherington, 1988, page 89], and Etherington proved that the procedure always converges for finite ordered network theories, and in a single pass for normal theories. We've modified the procedure to create a deterministic algorithm which finds some extension of any finite, disjunction-free ordered theory in a single pass, as we'll show in the forthcoming paper [Kautz and Selman, 1989]. The basic idea is to use the literal ordering to induce a total ordering on the set of default rules, so that an extension can be built incrementally from W by always applying the lowest applicable rule. This deterministic algorithm finds a particular, arbitrary extension. It can be further modified to iteratively compute all possible extensions of a theory, although some patience on the part of the user may be required: even a disjunction-free ordered theory may have an exponential number of extensions.

Unary theories do not share this low complexity.

Theorem 2 *Finding an extension of a unary default theory (or determining that it has no extension) is NP-Hard.*

The proof of this theorem depends on the following reduction of formulas in 3CNF to default rules.

Definition: σ, $\aleph$ and $\aleph'$
We will use σ to stand for a propositional 3CNF formula. $\aleph$ is the set of set of atoms which appear in σ. $\aleph'$ contains $\aleph$ plus a new atom p' for every atom p in $\aleph$.

Definition: π and η
The functions π and η map literals from $\aleph$ to $\aleph'$ as follows: π maps each positive literal to itself, and each negative literal $\overline{p}$ to a p'; while η maps each positive literal p to p', and each negative literal $\overline{p}$ to p. Thus,

$$\pi(p) = p \qquad \pi(\overline{p}) = p'$$
$$\eta(p) = p' \qquad \eta(\overline{p}) = p$$

Definition: f_U
The function f_U maps a formula σ in 3CNF to the union of the following three groups of default rules:

(A) For each atom p which appears in σ, where $p' = \pi(p)$, the following two rules appear:

$$\frac{:p}{p} \qquad \frac{:\overline{p}}{\overline{p}}$$

(B) Likewise for each p, the following two rules appear:

$$\frac{p : \overline{p'}}{\overline{p'}} \qquad \frac{:p' \wedge \overline{p}}{p'}$$

(C) For each clause $x \vee y \vee z$ of σ, the following three rules appear, where F_{xy}, F_{xyz}, $\mathcal{F}$, and $\mathcal{Z}$ are new atoms:

$$\frac{\eta(x) : F_{xy} \wedge \overline{\pi(y)}}{F_{xy}} \qquad \frac{F_{xy} : F_{xyz} \wedge \overline{\pi(z)}}{F_{xyz}}$$

$$\frac{F_{xyz} : \mathcal{F} \wedge \overline{\mathcal{Z}}}{\mathcal{F}}$$

Definition: f_{UE}
The function f_{UE} maps a 3CNF formula σ to the union of $f_U(\sigma)$ and the set containing the single default rule

$$\frac{\mathcal{F} : \mathcal{Z}}{\mathcal{Z}}$$

Proof of Theorem 2 *(sketch)* An arbitrary 3CNF formula σ is satisfiable if and only if the theory containing just the rules $f_{UE}(\sigma)$ has an extension. Therefore the problem of determining if a unary theory has an extension is also NP-Hard.
(if) Suppose $f_{UE}(\sigma)$ has an extension $\mathcal{E}$. By the rules in group (A) every atom in σ or its negation appears in $\mathcal{E}$. The rules in group (B) ensure that any extension containing p contains $\overline{p'}$, and any

containing $\overline{p}$ contains p'. Furthermore, it must not be the case that $\mathcal{F} \in \mathcal{E}$; because rule (D) would make $\mathcal{Z} \in \mathcal{E}$, so no rule in group (C) could ground $\mathcal{F}$. Thus no triple of group (C) rules corresponding to any one clause in σ can all apply in $\mathcal{E}$. Therefore every clause in σ is satisfied by any model which agrees with $\mathcal{E}$ on the truth value of every atom in $\aleph$.

(only if) Suppose M is a model of $\aleph$ which satisfies σ. Let $\mathcal{E}$ be the set of literals which hold in M, together with p' or $\overline{p'}$ for every literal $\overline{p}$ or p respectively which holds in M. Then $\mathcal{E}$ is an extension of $f_{UE}(\sigma)$. Note that $\mathcal{E}$ is grounded by the rules in groups (A) and (B), and that none of the rules in groups (C) or (D) apply. ■

The inherent complexity of unary theories is surprising, particularly in light of their close resemble to languages such as NETL [Fahlman, 1979] and the graphs which have been used to represent inheritance hierarchies [Etherington and Reiter, 1983]. One reason that their complexity has gone unnoticed may have been due to the concentration on problems which could be represented by acyclic graphs. Such problems fall in the class of ordered theories, which, by the previous theorem, are tractable.

A typical inheritance problem is to find a consistent set of default properties for a single individual (by default, "Tweety"). This can be done in the default logic formalism by having W contain a single atom which stands for the existence of the individual in question. The literals in the expansion then stand for the properties the individual inherits. This kind of reasoning is similar to what Touretzky calls "upward, credulous, decoupled" reasoning [Touretzky *et al.*, 1987]. They differ in that an "expansion" of a Touretzky-style hierarchy always contains all inheritance paths for all individuals and classes. The results of an decoupled reasoner, however, can be paralleled in default logic by letting W iterate through the set of all atoms, and finding an expansion based on each one individually.

Although ordered theories are still quite expressive, some natural situations do map into unordered theories. Consider the "corrupt city government" example illustrated in figure 4. We are using default rules to represent the concept "most". This year, most Republican councilmen are running for office, as are most Democratic councilmen. Furthermore, most councilmen running for office are under indictment. The District Attorney is Democratic, and will push the cases against the Republicans much harder than the cases against the Democrats. Therefore most Republican Councilmen who are under indictment are *not* running for office. This final condition is most naturally represented by a justification on the default rule for Republicans running for office, that is:

$$\frac{\textit{Republican} : \textit{Running} \wedge \overline{\textit{UnderIndictment}}}{\textit{Running}}$$

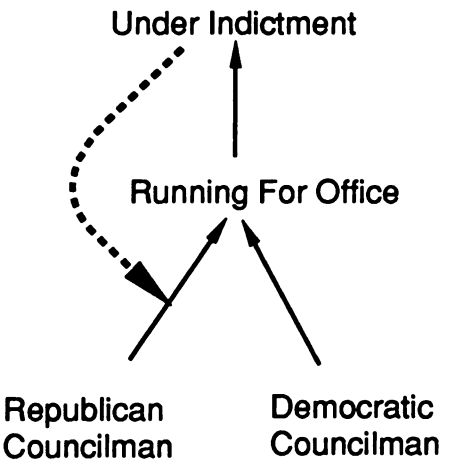


Figure 4: An unordered default theory.

It is not possible, as some might think, to instead make "not Republican" a justification on the "under indictment" rule, which would leave the theory ordered. It is easy to verify that there are worlds where most Republicans who are running for office are under indictment, and yet most Republicans who are under indictment are not running for office.

The complexity of general unary theories is also of significance due to their close relation to truth or reason maintenance systems. Unary theories are less expressive than any current TMS or RMS, and the negative result should carry over: that is, such a system could require an exponential amount of time to achieve consistency.

The intuitive cause of the complexity of unordered theories appears to be the ability to make default assumptions which must later be retracted, without the addition of new information. Figure 5 summarizes the results of this section.

6 The Complexity of Goal-Directed Reasoning

An extension can be thought of as one way the world could be, which is consistent with all of our knowledge of default information. If a theory happens to have a unique extension, and the extension is reasonably small, then it is practical to take the aim of default inference to be the computation of that extension. But a theory may have many extensions, and it may appear wasteful to compute an entire extension when one is only concerned with the truth value of a handful of propositions. Thus another way to frame a default reasoning problem is to ask if a given proposition is

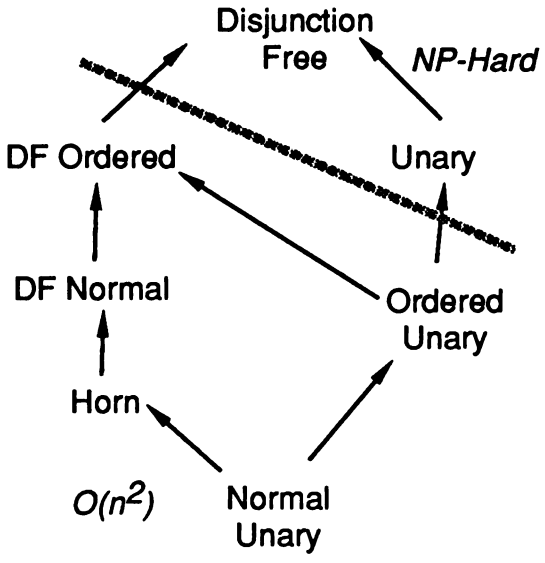


Figure 5: Finding an extension.

true in some extension, or perhaps in all extensions.

We'll call the problem of determining whether a given proposition holds in any extension "goal-directed" reasoning. This corresponds to asking, for a given p, whether there is "good" evidence for p, that is, evidence that is not overridden by other information. If p is desirable, then this kind of reasoning is appropriate for a very optimistic reasoner (p could well be true, therefore I'll assume it is); if p is undesirable, then it is appropriate for a very cautious reasoner.

Reiter [Reiter, 1980] showed that p holds in some extension of a normal theory just in case there is a top-down default proof of p. (A top-down default proof is, roughly, a sequence of non-default proofs; the first proves the goal given W and the conclusions of some set of the default rules; the next proves the antecedents of those defaults, perhaps given the conclusions of another set of of default rules; and so on, until a proof which only depends on W is reached.) As we noted above, Touretzky's notion of "credulous" reasoning is similar to finding an extension; he has no notion similar to goal-directed reasoning.

Goal-directed reasoning is much more difficult than simply finding an extension, as the following theorems demonstrate.

Theorem 3 *Determining if a given literal p appears in any extension of a propositional, disjunction-free normal theory is NP-Hard.*

Theorem 4 *Determining if a given literal p appears in any extension of a unary ordered theory is NP-Hard.*

Definition: f_{UG}
The function f_{UG} maps a 3CNF formula σ to the union of $f_U(\sigma)$ and the set containing following rule, where $\mathcal{T}$ is a new atom:

$$\frac{:\overline{\mathcal{F}} \wedge \mathcal{T}}{\mathcal{T}}$$

Proof of Theorem 4 *(sketch)* An arbitrary 3CNF formula σ is satisfiable if and only iff $\mathcal{T}$ holds in some extension of the theory containing just $f_{UG}(\sigma)$. This is because $\mathcal{T}$ can only appear in an extension in which $\mathcal{F}$ does not appear. Note $f_{UG}(\sigma)$ is ordered, unlike $f_{UE}(\sigma)$. ■

These theorems show that backward, goal-directed default reasoning is much harder than forward-directed reasoning. Ordering the theory does not reduce complexity; difficulties do not arise just from the possibility of one rule "undercutting" the support for another, but from the way in which defaults can be used to make a set of non-deterministic choices about the truth-values of various propositions. Searching for an extension where some proposition holds simulates the action of an oracle in a non-deterministic computer, which selects a successful computation from the set of possible computations. Upon reflection, then, the NP-hardness results are to be expected.

There is a useful default system, however, which does not fall prey to these negative results.

Theorem 5 *There is an $O(n)$ algorithm which determines if a given literal p appears in any extension of a horn default theory, where n is the number of default rules in the theory.*

This algorithm is based on the fast "pebbling" algorithm for determining satisfiability of a set of propositional horn clauses in ordinary logic [Dowling and Gallier, 1984]. If the given p is a positive literal, one translates only the default rules with positive conclusions into horn clauses, and then runs the pebbling algorithm. If p is negative, then one also includes those defaults whose conclusion is p. This fast algorithm is possible because negative conclusions can only *block* the application of another rule, but never enable another rule.

Horn default theories may have some practical applications in artificial intelligence. Such a theory is like a variable-free "pure" Prolog program (that is, no negation by failure), where all the rules are defaults. A much more interesting system would allow W to contain (non-default) horn clauses, instead of simply a set of literals, so both default and non-default information could be represented. We are currently investigating the complexity of such a system.

Figure 6 summarizes the results of this section.

7 The Complexity of Skeptical Reasoning

In a skeptical reasoning problem one asks if a given proposition holds in all extensions of a theory. One should note that skeptical reasoning cannot be derived

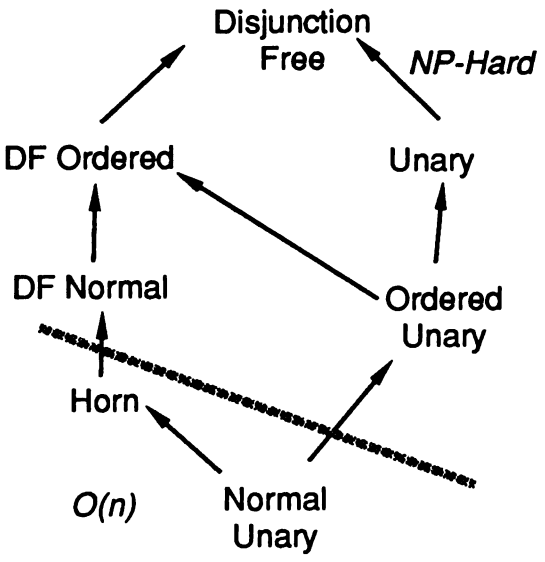


Figure 6: Goal-directed reasoning.

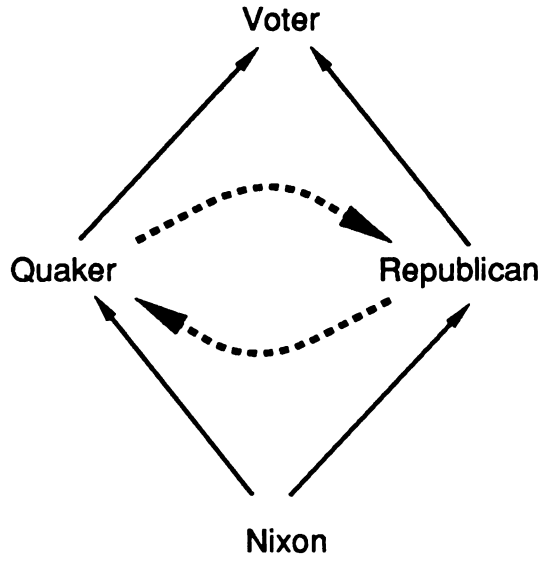


Figure 7: The extended Nixon diamond.

from goal-directed reasoning; that is, one cannot skeptically affirm p if there is no default proof of $\overline{p}$. This is because some extensions may contain neither p nor $\overline{p}$.

Skeptical reasoning is often taken to be the central aim of default logic. It possesses a number of attractive formal properties absent from the arbitrary-extension and goal-directed patterns. For example, the set of skeptical conclusions of a theory is closed under ordinary logical deduction. This leads to the practical advantage of allowing a certain amount of (interaction-free) decomposition in solving problems expressed in the logic. Finally, skeptical reasoning comes closest to providing a simplified version of probabilistic reasoning. For example, one cannot skeptically conclude both p and $\overline{p}$, just as one cannot assign a probability greater than 0.5 to both p and $\overline{p}$.

Our use of the term "skeptical" is closely related to Touretzky's. He takes the output of a skeptical reasoner to be the set of inheritance paths for every term in the hierarchy. When applied to inheritance hierarchies, our formulation of skeptical reasoning only determines if a single property of a single individual or class holds in every extension. It is a simple matter to do a double iteration, however, on the initial atom in W and the literal p under consideration. The result would be a complete set of properties inherited by each individual or class in every extension. Since this iteration would require n^2 calls to the default logic skeptical reasoner (where n is the number of literals), either definition of what constitutes a skeptical solution to a particular class of problems leads to the same categorization in terms of polynomial or NP-hard problems.

As one would expect from the results of the previous section, skeptical reasoning is hard for unary ordered theories.

Theorem 6 *Determining if a given literal p appears in every extension of a unary ordered theory is co-NP-Hard.*

Proof *(sketch)* An arbitrary 3CNF formula σ is unsatisfiable if and only iff $\mathcal{F}$ holds in all extensions of the theory containing just $f_U(\sigma)$. ∎

In fact, skeptical reasoning is strictly harder than simple goal-directed reasoning. While goal-directed reasoning is easy for horn theories, as [Kautz and Selman, 1989] will show, skeptical reasoning is not.

Theorem 7 *Determining if a literal p appears in every extension of a horn default theory is co-NP-Hard.*

Lemma 8 *There are default logics for which it is strictly harder to determine if a given literal p appears in all extensions than to determine if it appears in some extension of a theory.*

The lack of progress in devising complete and tractable algorithms for any kind of skeptical reasoning has lead some researchers to suppose that any formulation of reasoning based on an intersection of extensions or expansions is intractable. An example sometimes used to demonstrate this point is the "extended Nixon diamond", as shown in figure 7. Nixon inherits from "Voter" in all three extensions, but through Republican in one, Quaker in the other, and both in the third. (Note that in the default logic formulation, unlike Touretzky's system, no special priority is given to the default rules that lead directly out of a leaf node such as Nixon.)

This problem and others like it can be encoded entirely in a normal unary theory. We have devised a

```
normal-unary-skeptical(p_k, D)
    input: p_k & set D of normal unary defaults
    output: "yes" iff p_k holds in all extensions

    M := {p_1, p_2, ..., ¬p_k, ..., p_n}
    while [ pos-consistent(M) ] do
        if [ exists q, ¬r in M such that
                (NOT grounded(q, M) OR
                neg-inconsistent(q, ¬r, M)) ]
            then M := (M − {¬q}) ∪ q
            else return "no"
        fi
    od
    return "yes"
end.
```

Auxiliary definitions:

fixed-pos(q, M) iff
- a) $: q/q$ in D,
- b) $: \neg q/\neg q$ not in M, and
- c) $\neg r$ in M, for each rule $r : \neg q/\neg q$ in D

pos-consistent(q, M) iff
- for all q, fixed-pos$(q, M) \supset q$ in M

grounded(q, M) iff
- exists a sequence $q_0, q_1, q_2, \ldots, q_k = q$
- for some q_0 with
  - a) q_i in M, $0 \leq i \leq k$,
  - b) $: q_0/q_0$ in D, and
  - c) $q_{i-1} : q_i/q_i$ in D, $0 \leq i \leq k$

neg-inconsistent$(q, \neg r, M)$ iff
- a) q and $\neg r$ in M
- b) $q : r/r$ in D,
- c) $: \neg r/\neg r$ not in D, and
- d) $\neg s$ in M, for each rule $s : \neg r/\neg r$ in D

Figure 8: Unary normal skeptical reasoning algorithm.

sound and complete polynomial algorithm for skeptical reasoning in this logic.

Theorem 9 *There is an $O(n^3)$ algorithm which determines if a given literal p appears in every extension of a normal unary theory, where n is the number of default rules in the theory.*

The main body of the algorithm (for the case where W is empty, and p is positive) appears in figure 8, with the complete algorithm and proof of correctness in [Kautz and Selman, 1989]. To determine if p holds in all extensions, the algorithm attempts to construct a model containing $\overline{p}$ which is consistent with some extension. To our knowledge, this is the first time a sound and complete skeptical algorithm has been discovered. One must remember, however, that this default system does not include any kind of priority among rules (such as inferential distance).

The reader may gain some understanding of the algorithm by "running" it on the extended Nixon diamond example. The set of rules D, where each proposition is abbreviated by its initial letter, is:

$$\frac{n : r}{r} \qquad \frac{n : q}{q} \qquad \frac{r : \overline{q}}{\overline{q}} \qquad \frac{q : \overline{r}}{\overline{r}}$$

$$\frac{r : v}{v} \qquad \frac{q : v}{v}$$

Rather than including n in W, we'll simply add a default rule which always adds n. Since no rule adds $\overline{n}$, this yields exactly the same set of extensions.

$$\frac{: n}{n}$$

We wish to determine if p holds in all extensions. The model M is initially set to

$$M_1 = \{n, r, q, \neg v\}$$

M_1 is positive consistent, and all its elements are grounded. However, $(r, \neg v)$ is negative inconsistent, because a rule with precondition r adds v, and no rule whose precondition holds in M adds $\neg v$. So r is replaced by $\neg r$, yielding the next version of M:

$$M_2 = \{n, \neg r, q, \neg v\}$$

Now the algorithm may notice that $(q, \neg v)$ is negative inconsistent, so q is replaced.

$$M_2 = \{n, \neg r, \neg q, \neg v\}$$

But now $(n, \neg r)$ is negative inconsistent, so n must be replaced by $\neg n$. (Due to the random choice made in the **if** statement, this pair could have also been chosen in the previous step.)

$$M_3 = \{\neg n, \neg r, \neg q, \neg v\}$$

M_3 is not positive consistent, because n is fixed positive. Therefore the algorithm returns "yes", v holds in all extensions.

Figure 9 summarizes the results of this section.

8 Conclusions

Some of the main intuitions gained from this study follow.

- It is easy to determine a maximal consistent set of conclusions of a non-monotonic theory if the rules can be so ordered so that the justification of a rule never makes reference to a literal which could be affected by a rule which appears later in the ordering.
- Forward-chaining default reasoning is easier than backward-chaining, goal-oriented default reasoning. It may be practical to use a system based on default logic to vivify (fill out) an incomplete

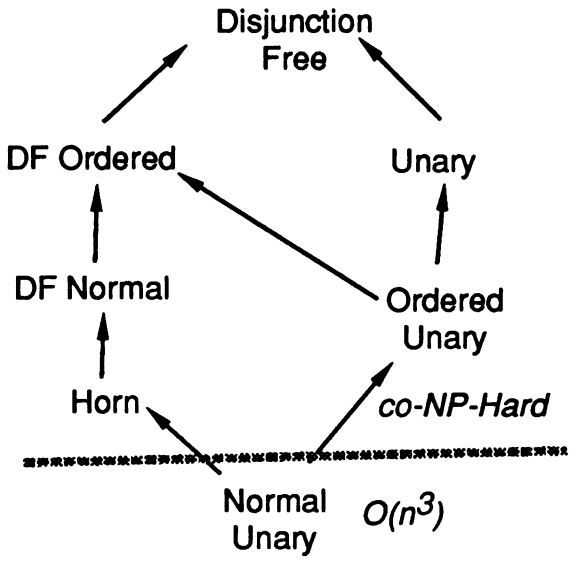


Figure 9: Skeptical reasoning.

knowledge base, by forward chaining to an arbitrary extension [Etherington *et al.*, 1989b]. It is probably impractical, however, to use default logic as extension to normal resolution-style theorem proving.

- It's usually a bad idea to quantify over extensions, either existentially or universally. If possible, try to reformulate the problem so that there is a unique extension, or the differences between extensions do not matter.
- Despite these warnings, there are some special and useful cases of default reasoning which can be efficiently handled (e.g., goal-directed reasoning for horn defaults and skeptical reasoning for normal unary defaults); don't take the complexity of any problem involving default reasoning for granted.

References

[Dowling and Gallier, 1984] William F. Dowling and Jean H. Gallier. Linear time algorithms for testing the satisfiability of propositional horn formula. *Journal of Logic Programming*, 3:267–284, 1984.

[Etherington and Reiter, 1983] David W. Etherington and Raymond Reiter. On inheritance hierarchies with exceptions. In *Proceedings of AAAI-83*, 1983.

[Etherington *et al.*, 1989a] D. Etherington, K. Forbus, M. Ginsberg, D. Israel, and V. Lifschitz. Critical issues in non-monotonic reasoning. In *Proceedings of Knowledge Representation and Reasoning '89*, Toronto, Canada, 1989.

[Etherington *et al.*, 1989b] David W. Etherington, Alex Borgida, Ronald J. Brachman, and Henry Kautz. Vivid knowledge and tractable reasoning: Preliminary report. Submitted for publication, 1989.

[Etherington, 1988] David Etherington. *Reasoning with Incomplete Information.* Morgan Kaufman, 1988.

[Fahlman, 1979] S. E. Fahlman. *Netl: a System for Representing and Using Real-World Knowledge.* M.I.T. Press, 1979.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability.* W.H. Freeman and Company, 1979.

[Kautz and Selman, 1989] Henry A. Kautz and Bart Selman. Hard problems for simple default logics (expanded version). In preparation, 1989.

[Levesque, 1986] Hector Levesque. Making believers out of computers. *Artificial Intelligence*, 30(1):81, October 1986.

[McCarthy, 1985] John McCarthy. Applications of circumscription to formalizing common sense knowledge. In *Proceedings from the Non-Monotonic Reasoning Workshop.* AAAI, October 1985.

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(2), April 1980.

[Selman and Kautz, 1988] Bart Selman and Henry Kautz. The complexity of model-preference default theories. In *Proceedings of CSCSI-88*, June 1988. (also in: *Lecture Notes in Artificial Intelligence 346: Non-Monotonic Reasoning*, Springer-Verlag, 1989).

[Touretzky *et al.*, 1987] D. S. Touretzky, R. Thomason, and J. F. Horty. A clash of intuitions: the current state of nonmonotonic multiple inheritance systems. In *Proceedings of IJCAI-87*, pages 476–482, Milan, Italy, 1987.

Localizing Temporal Constraint Propagation*

Johannes A. G. M. Koomen
Department of Computer Science
University of Rochester
Rochester, New York

Abstract

Allen's interval-based temporal logic [1983] derives some of its power from the well-defined way constraints can be propagated through a network of interval relations. However, the cubed time and quadratic space complexity in a fully connected network is prohibitive. This paper demonstrates that Allen's proposed solution, grouping intervals into clusters and propagating constraints within clusters only, is inadequate. Instead, an automatic reference method is proposed which creates a reference hierarchy based on temporal containment, dynamically restructures the hierarchy as more assertions are added to the network, and obviates the need to define an a priori reference structure. Using this hierarchy, most of the relations between disjoint intervals can be eliminated without loss of information, considerably reducing the connectivity of the network, and therefore the cost of constraint propagation. Empirical results on both random and planner-generated networks bear out the superior performance of the automatic referencing method.

1 Introduction

Interval-based temporal logic, introduced by Allen [1983], has proven to be a powerful paradigm. In many A.I. systems, information about the temporal relation between two events or propositions is as important, if not more so, as absolute length or position on a time line of any one such event or proposition. Recognizing that intervals can be related to other intervals in at most seven different ways (before, meets, overlaps, starts, during, finishes and equals) plus their inverses, Allen proposed a way of storing partial temporal knowledge by maintaining sets of possible temporal relations between any two intervals, and provided a temporal constraint propagation algorithm. Knowledge can be added to a network of interval constraints by reducing the set of possible temporal relations between two intervals, and propagating the effects of this reduction to all other intervals related to these two.

Given the relational constraint sets between intervals i and j and between j and k, the relational constraint set between i and k can be obtained through a transitivity function which successively indexes a transitivity table by the elements of the i–j and j–k sets and computes the union of the table entries. If there already existed a set of constraints between i and k, this set is intersected with the computed union to arrive at a new constraint set. If the cardinality of the resulting set is 1, then the constraint between i and k has been uniquely determined; if the intersection resulted in the empty set, then a conflict exists in the network.

Since the cardinality of the initial sets of possible relations is at most 13, each relational constraint can be asserted and updated at most 12 times.[1] However, propagating the effects of such an update is proportional to the branching factor, *i.e.*, the number of other intervals related to the two intervals whose constraint changed. Without imposing structure, a network of temporal relations between n intervals tends toward full connectivity. Hence, for each relational constraint added, Allen's basic algorithm attempts to propagate the effect to all other "Comparable" (*i.e.*, connected) intervals, $2n - 2$ times invoking the (expensive!) transitivity function Constraints. Each time propagation reduces another constraint, this reduction too will be propagated across the network. Vilain and Kautz [1986] showed that the time complexity of this algorithm in terms of Constraints invocations is $O(n^3)$. As the number of intervals in the network grows, this becomes entirely insufferable. Moreover, in a system using the interval logic, activity tends to focus on the

*This work was supported by NSF research grant DCR-8351665. We thank the Xerox Corporation University Grants Program for providing equipment used both in the implementation of the algorithms and in the preparation of sections of this paper.

[1] Asserting the set of all 13 possible relations is equivalent to having no information at all.

most recently added intervals; new information does not affect old relations much, and propagating effects back to older intervals tends to be useful only for consistency checking.

Clearly, the cubed time and quadratic space complexity for a temporal database of this nature is prohibitive for all but the smallest problem sets. Allen proposed the use of reference intervals to limit constraint propagation, thereby reducing the average time and space requirements. This paper first demonstrates that, in general, arbitrary reference clusters fail to propagate crucial constraints, resulting in an underconstrained network. Then a new algorithm is presented which allows a temporal database to dynamically create, without loss of information, a reference hierarchy based on temporal containment. Directly stored relations between disjoint intervals can be eliminated if the same relation can be found along a path through the containment hierarchy. As such a hierarchy tends to be tree-like, this search is acceptable (*i.e.*, logarithmic). Empirical results of experiments based on random networks of intervals and on a more realistic planning problem bear out the superior performance of this algorithm.

2 Localizing propagation

Allen proposed adopting a notion of reference intervals similar to that of Kahn and Gorry [1977] to reduce the space requirements of an interval network. Clusters are formed within the network by associating intervals with one or more reference intervals. Constraint propagation takes place within each cluster only, while inter-cluster relations are obtained indirectly by applying the transitivity function along paths through the network, where paths consist of reference intervals only. For instance, consider the two activities *gardening* and *dinner* with associated intervals Gardening and Dinner. The intervals associated with all activities relating to this instance of gardening, such as Mowing and Raking, would be related directly to Gardening, and likewise, intervals such as BoilWater, CookFood, *etc.* would be directly related to this instance of preparing for and having dinner. The intervals BoilWater, CookFood, *etc.* would not be directly related to Gardening or any of its subintervals, however; those relationships would be determined by applying the constraint propagation algorithm along the path through Dinner. The effect of further constraining two dinner-related intervals would not be propagated to any gardening-related intervals. This mechanism obviously can reduce the branching factor at each node considerably, resulting in a significant space savings (fewer relations to store) as well as a speed-up in adding new constraints due to propagation to fewer related intervals. The time to fetch inter-cluster relations is increased, but this can be kept manageable (*i.e.*, logarithmic) if the reference intervals are organized in a tree-like hierarchy.

Allen claims that imposing a reference hierarchy "can be done with little loss of information," as long as "care is taken." He does not make clear, however, of what this care consists. The fact is that it is rather easy to lose information due to the structure of the reference hierarchy, particularly between intervals in different, temporally overlapping clusters. To flesh out the example above, assume that the following constraints have been asserted:

c_1: Mowing {*starts*} Gardening
c_2: Mowing {*before*} Raking
c_3: Raking {*finishes*} Gardening
c_4: BoilWater {*starts*} Dinner
c_5: BoilWater {*meets*} CookFood
c_6: CookFood {*before, meets*} EatFood
c_7: EatFood {*finishes*} Dinner
c_8: Dinner {*meets*} Dishes

Now suppose I decided to boil the water while I'm raking the lawn:

c_9: Raking {*equals*} BoilWater

In a "flat" network, the following additional constraints, among others, would be derived automatically:

c_{10}: Gardening {*overlaps*} Dinner
c_{11}: Mowing {*before*} Dinner
c_{12}: Mowing {*before*} Dishes
c_{13}: CookFood {*during*} Dinner

However, consider a structured system where Gardening is a reference interval for Mowing and Raking, and Dinner is a reference interval for BoilWater, CookFood and EatFood. The reference structure is depicted in Figure 1.

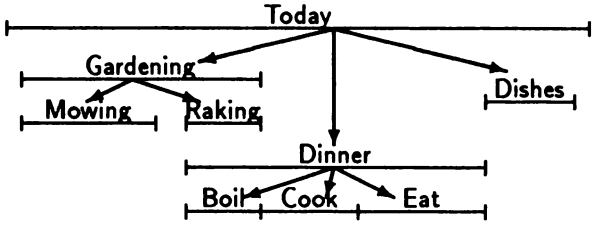


Figure 1: Example of interval reference structure

In this case constraint c_{10} would be derived, but the interval Mowing would not be comparable with Dinner and Dishes (they do not share a reference interval, nor have explicit relationships been asserted between them), and hence constraints c_{11} and c_{12} would not be derived. Unfortunately, computing the constraint between Mowing and Dinner along the path Mowing–Gardening–Dinner results in the constraint {*before, meets, overlaps*} which is clearly weaker than constraint c_{11} obtained in the flat system. Note that over-

lapping reference intervals and multiple reference intervals per simple interval are by no means contrived situations; for instance, they can easily arise in planning situations [Allen and Koomen, 1983]. Moreover, this example shows that merely requiring that intervals are wholly contained in their reference intervals, as Allen suggests, is insufficient to assure proper inferencing. Allen already noted that merging two reference subtrees whenever such an overlap is detected tends to quickly flatten the entire reference hierarchy, defeating the purpose for reference intervals.

3 Automatic referencing

Devising a reference structure for a temporal constraint network such that no information is lost turns out to be a very nasty problem, and certainly one that a temporal reasoning system ought not to leave up to higher level reasoning systems, as Allen suggests. It is hard, if not impossible, to predict what effect predefining a reference structure has on temporal inferencing, and it certainly requires taking into account the order in which constraints are posted as well as detailed knowledge of the temporal constraint propagation algorithm, details of which would best be kept in the specialized temporal reasoning system. It would be very desirable if the temporal reasoning system itself could structure the network dynamically on the basis of posted and derived constraints, and do it without losing information. This and the following sections describe a method for doing this.

As Allen [1983] and Kahn and Gorry [1977] suggested, a reference hierarchy is naturally based on temporal containment: higher level intervals have a greater span in time, and lower level intervals are contained in higher ones. Propositional inheritance is an obvious reason for basing the structure on containment: any proposition that holds over an interval X necessarily holds over any subinterval of X, and any relation between X and another, disjoint interval Y, is necessarily true between any subinterval of X and Y. Another reason a reference structure based on temporal containment is desirable is that it reflects temporal locality of reference: most queries to a temporal database concern relations between intervals associated with propositions currently under consideration, such as planning a task, where propagation is likely to be useful, whereas very few queries would be concerned with the constraints between two (semantically) unrelated and temporally distant propositions. However, this reasoning breaks down if a lot of global domain constraints apply to the problem at hand [Allen and Koomen, 1983]. For instance, in the blocks world a rule such as

$$\forall \text{ blocks } b_1, b_2, \text{ intervals } t_1, t_2$$
$$\text{ON}(b_1, b_2)@t_1 \ \wedge \ \text{CLEAR}(b_2)@t_2$$
$$\Longrightarrow \ \text{DISJOINT}(t_1, t_2)$$

would cause task–unrelated intervals to be temporally related, violating the notion of temporal locality of reference.

The automatic referencing method described in this paper revolves around two notions, namely, dynamically constructing a reference hierarchy based on containment, and reducing the interval branching factor by removing direct constraints that can be obtained indirectly through this reference hierarchy.

In the following discussion, the constraints between lower and higher level intervals in the reference hierarchy will be referred to as *uplinks* (with *downlinks* going the other way), and constraints between disjoint intervals as *sidelinks*, with the proviso that sidelinks are singular, *i.e.*, relational constraint sets with cardinality 1. Let Refs*(i) be the closure of Allen's Refs function, *i.e.*, the set of all reference intervals directly or indirectly above i. Let C(i,j) be the direct constraint between intervals i and j and N(i,j) be the constraint between i and j derivable from the hierarchy. (See [Koomen, 1988] for formal definitions of these links and more detailed descriptions of the automatic referencing algorithms.)

3.1 Constructing the hierarchy

The reference hierarchy is constructed on the basis of asserted uplinks and downlinks. If an uplink between intervals i and j is asserted (either explicitly or through propagation), *i.e.*, the new constraint set between i and j is a subset of {*starts,during,finishes*}, then j is a candidate reference interval for i. Adding a reference link from i to j, if necessary, is accomplished by he following algorithm:

```
to AddRef(i,j)
    if j ∈ Refs(i) then
      Exit
    else if j ∈ Refs*(i) then
      ;; i is already below j
      Refs(i) ← Refs(i) − {j}
    else begin
      Refs(i) ← Refs(i) ∩ {j}
      for each node k ∈ Refs*(j) do
        ;; remove now indirect links, if any,
        ;; from i to ancestors of j
        Refs(i) ← Refs(i) − {k}
      for each node h such that i ∈ Refs*(h) do
        ;; remove now indirect links, if any,
        ;; from descendants of i to j
        Refs(h) ← Refs(h) − {j}
    end
  end AddRef
```

In the gardening/dinner example above, for instance, the same hierarchy as depicted in Figure 1 is constructed dynamically as the constraints c_1, c_3, c_4 and c_7 are posted. Assuming that all gardening and dinner intervals are initially constrained to be during Today, constraint c_1 causes Gardening to become a reference

interval for Mowing while the reference link between Mowing and Today is removed. Likewise, the reference links between EatFood and Dinner (by assertion of c_7) and between CookFood and Dinner (by implication) are created, and the corresponding reference links to Today are removed.

3.2 Reducing the branching factor

The branching factor of an interval is reduced by removing singular[2] sidelinks and indirect uplinks with other intervals, provided those links can be derived from a path through the reference hierarchy. If a singular sidelink between intervals i and j is asserted (either explicitly or through propagation), *i.e.*, some interval is disjoint from another and the constraint set is singular, the procedure RemoveLink attempts to remove the sidelink. This is successful if and only if the same relation can be obtained by applying the constraint propagation algorithm along all paths through the reference hierarchy starting from i and ending up at j, and i and j do not have a reference interval in common.

```
to RemoveLink(i,j)
    if SingularP(C(i,j))
      and Refs(i) ∪ Refs(j) = ∅
      and C(i,j) = N(i,j)
    then Remove C(i,j) from the network
  end RemoveLink
```

If a singular uplink between intervals i and j is asserted (either explicitly or through propagation), *i.e.*, some interval is contained in another and the constraint set is singular, the uplink can be removed if it is an indirect uplink. Additionally, all singular sidelinks or indirect uplinks between i or any of its descendants and j or any of its ancestors may now be removable, since this uplink has added information to the hierarchy.

In the example above, the constraints c_{10}, c_{11} and c_{12} are all derived and recorded. Both c_{11} and c_{12} are sidelinks, so both are candidates for removal. There exists a path along which c_{12} can be obtained exactly, hence the relation c_{12} can be removed. However, there exists no reference path that obtains the same relation as c_{11} and therefore the relation c_{11} cannot be removed without loss of information. Note also that c_2 cannot be removed; in general, this is true for all relations between disjoint intervals that have a common reference interval.

4 Empirical results

The automatic referencing mechanism described above suffers no loss of information, since each relation is initially computed, and it is not deleted until it has been uniquely determined and it can be obtained indirectly through the reference hierarchy. This initial computation of a relation followed by its deletion may seem to be a lot of extra and unnecessary work—how have we gained anything? It turns out that the greatly reduced average branching factor at each node (and hence reduced propagation) more than offsets any extra work entailed in the automatic management of the reference hierarchy. Unfortunately, it is difficult to get a handle on the kind of improvement this mechanism affords, since it is very dynamic in nature, and therefore very much dependent on the order and kind of relations asserted.

Nevertheless, two experiments have borne out that this approach is generally superior in performance to the flat system. The first experiment consisted of generating a network of 101 intervals and randomly assigning singular relations between any two intervals until all pairs of intervals have singular relations, either posted or derived, that are locally consistent [Vilain and Kautz, 1986]. In all, 450 relations were posted to obtain the total of 5050 singular relations. The results are given in Table 1, which compares average values per node and total values of several statistics for the flat system (NoRefs) and the automatic referencing system (AutoRefs). Note in particular the sharply reduced average number of uplinks and sidelinks per node.

Statistic	NoRefs	AutoRefs
Branch factor, average	100.0	18.3
# UpLinks, average	19.4	1.4
# SideLinks, average	47.3	1.6
Reference depth, average	1.0	13.8
Propagations, total	775,015.0	252,411.0
Constraints invocations	1,105,346.0	475,585.0

Table 1: Propagation statistics on 101 randomly related intervals

The second experiment involved planning the task of exchanging the top two blocks of a stack using a two-armed robot, with the additional constraint that there is no room to place either block on the surface supporting the stack in an intermediate state (see [Allen and Koomen, 1983] for a more detailed description of this problem). The planner generates 33 intervals, asserting 65 temporal relations between them. In a flat system, another 1063 constraints are asserted due to constraint propagation, causing some 43000 invocations of the transitivity function Constraints and resulting in an average branching factor of 31, *i.e.*, the network ends up almost completely connected. Using the automatic referencing mechanism as described above, the average branching factor and the number of invocations of the transitivity function are roughly cut by a half. These experiments confirm our intuition that au-

[2]Non-singular uplinks or sidelinks should not be removed, as they are still subject to revision and should remain in the active network.

tomatic referencing should make things progressively better as the number of intervals increases.

5 Related issues

5.1 Non-monotonicity and backtracking

The automatic referencing algorithm, as well as the algorithms in Allen [1983], have been presented with the assumption of monotonic updates to the temporal database. Many reasoning systems, however, require either the ability to retract one or more previously posted constraints, or support of efficient backtracking. For a forward-chaining constraint propagation system this implies keeping both the derived constraints themselves and adequate dependency information. Hence, removing relations from the network as suggested in Section 3.2 is out of the question. Limited to $O(n^2)$ space, we can still benefit from the algorithm presented here by simply marking relations "broken" where they would have been removed. Propagation does not need to take place across broken links, and in addition, these links can still be used to obtain relational information, circumventing the need for traversing the reference hierarchy!

5.2 Propagating equality

Equality between intervals can occur frequently in a logic-based reasoning system. Intervals are associated with non-grounded terms which are said to hold over the intervals. In such a system, unification of terms requires unification of intervals. Temporal equality can be viewed as a form of temporal containment, and so should be usable for automatic referencing. The only obstacle is the fact that equality is symmetric, which begs the question: is it an uplink or a downlink? This can be resolved by assigning sequence numbers to intervals as they are introduced, indicating their age. Now the above mechanism for automatic referencing can be extended to include equality by redefining *uplink* to include *equals* if the link is between a younger and an older interval, making the older of two equal intervals a reference interval for the younger one. The definition of *downlink* is modified analogously. Using this extended definition increases the number of potential reference intervals, which in turn tends to reduce the overall network connectivity. Unfortunately, incorporating equality in the referencing mechanism like this will also have the effect of linearizing equivalence sets. Additional gain can be had by using a UNION-FIND algorithm to make the oldest interval in an equivalence set the reference interval for the set, and the youngest the reference interval for all subintervals.

6 Conclusions

Allen's interval-based temporal logic derives some of its power from the well-defined way constraints can be propagated through a network of interval relations. However, the cubed time and quadratic space complexity in a fully connected network is prohibitive. The solution proposed by Allen, namely to group intervals into clusters and to propagate constraints within clusters only, has been shown to be inadequate. Many relations obtained in a flat network are underconstrained in a clustered network when derived from application of the transitivity function along a path between clusters. The automatic reference method relieves the reasoning system using the interval-based temporal logic from having to design a reference structure. It automatically creates a reference hierarchy based on temporal containment, dynamically restructuring the temporal network. Using this reference hierarchy, most of the sidelinks, *i.e.*, singular relations between disjoint intervals, can be eliminated without loss of information, which considerably reduces the connectivity of the network, and therefore the cost of constraint propagation. Empirical results on both random and planner-generated networks bear out the superior performance of the automatic referencing method.

Acknowledgements

I thank James Allen and Josh Tenenberg for their encouragement, support, and comments on an earlier draft of this paper.

References

[Allen and Koomen, 1983] James F. Allen and Johannes A. Koomen. Planning using a temporal world model. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 741–747, Karlsruhe, W. Germany, 1983.

[Allen, 1983] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[Kahn and Gorry, 1977] K. Kahn and G. A. Gorry. Mechanizing temporal knowledge. *Artificial Intelligence*, 9(1):87–108, 1977.

[Koomen, 1988] Johannes A. G. M. Koomen. The TIMELOGIC temporal reasoning system. Technical Report 231, University of Rochester, Department of Computer Science, October 1988. (Revised).

[Vilain and Kautz, 1986] Mark Vilain and Henry Kautz. Constraint propagation algorithms for temporal reasoning. In *Proceedings of the 5th National Conference on Artificial Intelligence*, pages 377–382, Philadelphia, Pa., 1986.

Knowledge Representation in a Case-Based Reasoning System: Defaults and Exceptions

Phyllis Koton and **Melissa P. Chase***
The MITRE Corporation
Burlington Road
Bedford, MA 01730

Abstract

Case-based reasoning is a technique which draws inferences about a new instance presented to the system by comparing it to other instances previously presented to the system. The knowledge representation structure used by case-based reasoning systems is built up incrementally, with the structure modified each time a new instance is seen. In this paper, we formalize case-based inference and show that the conclusions that are made using this process are consistent with standard default reasoning.

1 Introduction

Case-based reasoning is a technique which draws inferences about a new instance presented to the system by comparing it to other instances previously presented to the system. The case-based approach to knowledge representation and reasoning has three underlying principles.

1. The ability to identify an instance as being *like* an instance already encountered is the fundamental means by which knowledge is applied.
2. Individual instances should be organized on the basis of their similarities and differences, in order to facilitate the identification of similar instances.
3. Aggregates of individual instances (called *generalizations*) form the basis for default reasoning.

These principles require that the knowledge representation system used by case-based reasoners differs from traditional knowledge representation systems in several ways. For example, whereas in traditional systems, new instances are added to the knowledge structure by the user, the case-based reasoner must decide where in its existing knowledge representation structure to integrate a new instance. In traditional systems, the principal relationship used to organize concepts is subcategorization (e.g., the `IS-A` or `AKO` link). In case-based reasoning systems, there is no distinguished link indicating subcategorization. It might seem that a knowledge representation structure organized in such a loose and dynamic fashion would not be conducive to deriving a coherent set of inferences. We show that exactly the opposite is true.

We begin with an informal description of the knowledge representation system used by case-based reasoning systems and an overview of the case-based reasoning process. The process is formalized in an algorithm for case-based inference. We then present a reformulation of case-based reasoning in terms of Reiter's default logic [Reiter, 1980], and use this to prove that the conclusions which can be drawn using case-based reasoning are valid and coherent.

2 The Case-based approach

The case-based approach centers around the *self-organizing memory system* developed by Kolodner [Kolodner, 1983, Kolodner, 1984]. A self-organizing memory system records and organizes instances or *cases*. Cases are made up of *features*, which can be thought of as attribute-value pairs. The memory system also creates generalization structures, or *gens*, which are structures that hold knowledge describing a group of similar cases. A gen is created from the similarities between the cases that it organizes. Individual cases that are stored in a particular gen are indexed by the features that distinguish them from the other cases in the same gen. Two cases are said to be *similar* if they are integrated into the same gen and share a set of differences with the gen. Each gen maintains a list of features that are common to most[1] of the cases it organizes. These are called the *norms* of the gen.

Following the scheme described in [Kolodner, 1983], the memory structure is represented as a discrimination net in which each node is either an individual case or a gen. Each pointer to a subnode is labeled (or *indexed*) by a feature of the subnode that differentiates it from the parent node. The features that differentiate the subnodes from the parent gen are called the

*This research was supported in part by the Defense Advanced Research Projects Agency.

[1] In many case-based reasoning systems, "most" is implemented as $\geq 2/3$.

diffs of the gen. Indexing requires two levels. The first level indicates the *category* of the index. The second level indicates the *values* that the feature takes on in the subnodes.

Figure 1 shows a fragment of a memory structure containing some knowledge about birds. The uppermost box, labeled $G0$, is a gen that has one norm, `covering = feathers`. The instances of things with feathers that have been presented to the memory structure can be differentiated by their feet, their diet, and their habitat. For example, things with feathers and clawed feet are described by gen $G1$, which contains additional norms that indicate that feathered, claw-footed things are likely to live on land and eat insects.

When presented with a new instance, a case-based reasoning system searches its memory for a similar case. The set of features which describe the new instance defines a set of paths through the memory structure. When installing a new instance in the case memory, one of three conditions obtains at each point in the path.

1. If exactly one case is stored at this point, the stored case and the new case are compared, their similarities (the norms of the new gen) are placed in a new gen, and they are indexed beneath the gen by their differences from each other. This is the way in which new gens are created by the memory.

2. If there is a gen at the point, the new case is indexed in the existing gen.

3. If no other case is stored at this point, the new case is simply installed there.[2]

The initial set of norms and diffs for a gen are determined by the features of the two original cases that create the gen. Subsequent cases added to the gen may not share all the norms. Eventually, a norm might no longer describe most of the cases stored in the gen. Kolodner calls this *overgeneralization*. Conversely, a feature that had initially been stored as a diff might turn out to characterize a large number of the cases subsequently stored in the gen and is no longer useful for differentiating among cases. Kolodner terms this *undergeneralization*. The memory structure contains mechanisms by which these conditions can be detected and corrected. For a further explanation of the mechanisms of the self-organizing memory structure, the reader is referred to [Kolodner, 1983].

[2]An input instance may have similarities with many previous cases. In a later section we show that this corresponds to the notion of multiple extensions.

3 An Algorithm for Case-based Inference

Having described the structure of the case memory, we now proceed to show how inferences are drawn using this structure. Given a new instance, the task of the case-based reasoner is to find the instances in the case memory most similar to the new one, and to reorganize the memory structure to account for the new information contained in the new instance. Finding similar instances allows the case-based reasoner to infer features that are not explicitly present in the new instance.

As stated in the previous section, the set of features which describe the new instance defines a set of paths through the memory structure. Starting from the root, the case-based reasoner uses the features of the new instance as "keys" to unlock the next level of nodes in the memory. Each level down in the memory structure represents a specialization of the gens preceding it. When the features of the new instance lead to a gen, the norms of that gen are assumed to hold for the new instance, unless they are overridden by more specific norms subsequently acquired from gens lower down in the memory structure. This can be formalized in the following way.

A case memory contains cases $C_1, \ldots, C_n$ described by feature sets $\phi_1, \ldots, \phi_m$, and gens $G_1, \ldots, G_r$ with norms sets $\nu_1, \ldots, \nu_s$ respectively. A feature is often represented as an attribute-value pair, $< a_i, v_i >$. A norm is also represented as an attribute-value pair.

The cases and gens are arranged in a discrimination net indexed by features of the cases in the network, with a distinguished node called the root. For each case or gen in the net, there exists at least one path from the root to that object.

Let $\mathcal{N}$ denote the sets of conclusions that can be inferred at any point P in the network. By "set of conclusions" we mean the features explicitly present at P and the features that are inherited. P can be either a gen or a case. There are k paths from the root to P and each path is of length l_k. Each path defines a set of conclusions at point P.[3] Formally,

$$\mathcal{N} = \bigsqcup_{i=1}^{k} \mathcal{N}_i, \quad \mathcal{N}_i = \bigsqcup_{j=1}^{l_k} \nu_j$$

where $\bigsqcup$ is defined as:

```
for each norm (a, v)
    if ∃x ∋ (a, x) ∈ N_i
    then N_i = (N_i − (a, x)) ∪ (a, v)
    else N_i = N_i ∪ (a, v)
next norm
```

[3]The interpretation of multiple paths to the same point is discussed in the next section.

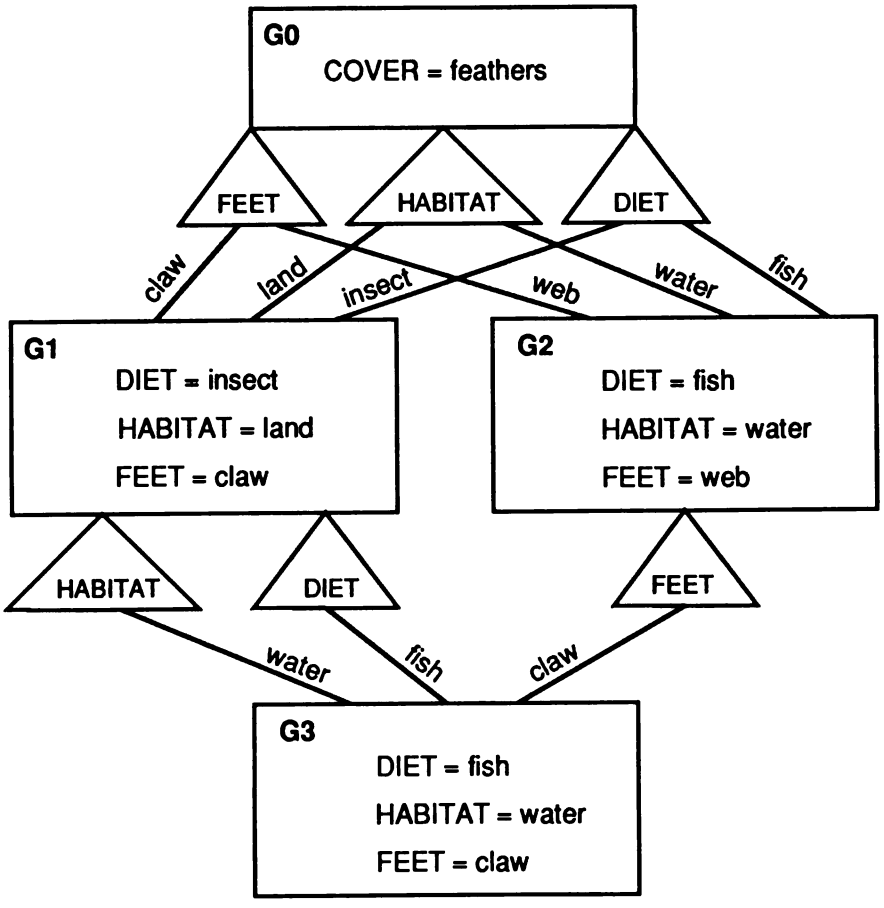


Figure 1: A fragment of memory structure.

In each step down the path, the new (more specific) value of the feature replaces the previous (more general) value, if such a value exists. This induces an order on the objects in the memory.

A gen G_j is said to be the most specific generalization along some path to a memory object, P_i, if it is the last gen along that path. The most specific generalization of a gen G is itself. The most specific generalizations of of a case C are the last gens on the path leading from the root to C.

4 Reformulating CBR in Terms of Default Logic

Reiter and Etherington[Etherington and Reiter, 1985, Etherington, 1987] have shown how various kinds of inheritance networks can be reformulated in terms of default logic[Reiter, 1980]. The motivation behind this endeavor is to provide a formal grounding for the prototypical reasoning carried out by these representations. In this section we will develop a similar correspondence between default theories and case memory. We will then be able to understand how case-based reasoning systems perform. We will also see that many of the desirable features of case-based reasoning systems, such as inferring default properties and reasoning about exceptions, emerge from the default logic reconstruction.

4.1 Default Logic

Recall that a *default theory* is an ordered pair, (D, W), consisting of a set of defaults D and a set of first-order formulae, W. A *default* is rule of inference of the form:

$$\frac{\alpha(\overline{x}) : \beta_1(\overline{x}), \ldots, \beta_m(\overline{x})}{\gamma(\overline{x})}$$

where $\alpha(\overline{x})$, $\beta_i(\overline{x})$, and $\gamma(\overline{x})$ are all formulae whose free variables are among $\overline{x} = x_1, \ldots, x_n$.

Intuitively this default rule can be interpreted: if $\alpha(\overline{x})$ is known and it is consistent to believe the $\beta_i(\overline{x})$, then one may conclude $\gamma(\overline{x})$. α is the *prerequisite*, the β_i are the *justifications*, and γ is the *consequent* of the default rule.

Defaults of the form

$$\frac{\alpha(\overline{x}) : \beta(\overline{x})}{\beta(\overline{x})},$$

are called *normal* defaults and default theories which contain only normal defaults are called *normal default theories*. Defaults of the form

$$\frac{\alpha(\overline{x}) : \gamma(\overline{x}) \wedge \beta(\overline{x})}{\gamma(\overline{x})},$$

are called *semi-normal* defaults and default theories which contain only semi-normal defaults are called *semi-normal default theories*. Most naturally occurring situations can be expressed solely in terms of these restricted kinds of defaults.

The sets of conclusions that are sanctioned by a default theory are called *extensions*. Default theories that have an extension are said to be *coherent* because they support an acceptable set of beliefs about an incompletely specified world. Although we can not completely characterize the default theories that possess extensions, some conditions that guarantee coherence are known. Reiter showed that all normal theories have at least one extension[Reiter, 1980]. Etherington has found some sufficient conditions that guarantee the coherence of semi-normal theories[Etherington, 1988]. Typically a default theory may have multiple extensions, although theories with unique extensions are preferable (since conclusions drawn from different extensions may be incompatible).

4.2 Reformulation of CBR

We will now establish a correspondence between case memories and default theories. Recall that the case memory is a discrimination network composed of generalizations and cases linked by diffs. In the following, we will assume that the case memory has the following structures. Each node in the memory (both cases and generalizations) has a unique label. This label is not necessarily a class name that can be used directly as an assertion about class membership, although case memories that include class names and IS-A links between classes can be constructed. Each node has a list of features. Each feature has an attribute name and an associated value. If the node is a case these features represent the properties possessed by the case; if the node is a gen, they represent the properties possessed by most cases organized by the gen. Each diff is a link from a gen to a gen or a case. Each diff is labeled by a feature that differentiates the more specific node from the generalization, i.e., the gen and the node have different values for this attribute. Note that there may be multiple diffs linking two nodes.

A case memory can be translated into default logic as follows. We always assume that an individual is an instance of the root of the case memory, so W always contains the formula $\forall x.Root(x)$. Since all properties represented in the memory are contingent upon the cases that have been presented, there are no first-order formulae relating features and nodes. However, if the values of an attribute are mutually exclusive, W may contain formulae of the form $\forall x.f(x,a_i) \supset \neg f(x,a_j)$ for each $a_i \neq a_j$. The features of nodes (i.e., the norms of generalizations and the features of cases) and the diffs that link generalizations to nodes provide the "inference" rules of the case based reasoner and are translated into default rules as follows:

1. **Feature.** A node, $N(x)$, has a feature $f(x,a)$ and there are no diffs at this node corresponding to this feature. This is interpreted to mean: normally N's have value a for feature f. Identified with:

$$\frac{N(x) : f(x,a)}{f(x,a)}$$

2. **Link.** A generalization, $G(x)$, is linked to a node, $N(x)$, by a diff $f(x,b)$. This is interpreted to mean: N's differ from G's by possessing value b for feature f. Identified with:

$$\frac{N(x) \wedge f(x,b) : N(x)}{N(x)}$$

3. **Differentiated Norm (DN).** A generalization, $G(x)$, has a norm $f(x,a)$ and diffs $f(x,b_1),\ldots,f(x,b_n)$. This is interpreted to mean: G's usually have value a for feature f, unless there are exceptions. Identified with:

$$\frac{G(x) \wedge \neg f(x,b_1) \wedge \cdots \wedge \neg f(x,b_n) : f(x,a)}{f(x,a)}$$

4. **Restricted Closed World (RCW).** A generalization, $G(x)$, has a norm $f(x,a)$, is linked to a node $N(x)$ by the diff $f(x,b)$, and has additional diff links to $N(x)$, $f_1(b_1),\ldots,f_n(b_n)$. This is interpreted to mean: when G can not be specialized to an N by any other diffs from G to N, do not assume G has value b for feature f. Identified with:

$$\frac{G(x) \wedge \neg f_1(x,b_1) \wedge \cdots \wedge \neg f_n(x,b_n) : \neg f(x,b)}{\neg f(x,b)}$$

When $f(x,b)$ is the only diff linking G and N, the default rule becomes:

$$\frac{G(x) : \neg f(x,b)}{\neg f(x,b)}$$

Using these defaults we can translate the fragment of case memory depicted in Figure 1 into a default theory. The corresponding default theory is shown in Figure 2.

G_0 is the root of the memory, and corresponds to the generalization of all the cases that have been integrated into the memory. The only feature common to all the cases is that they are covered with feathers. G_1 corresponds to the generalization of birds that have clawed feet, whose habitat is land, and whose diet consists of insects. G_2 corresponds to the generalization of birds that have webbed feet, whose habitat is water, and whose diet consists of fish. G_3 corresponds to the generalization that is a specialization of both G_1 and G_2. G_3 differs from G_1 in representing those cases that are like G_1's but whose habitat is water and diet is fish; it differs from G_2 in representing those cases that are like G_2's but have clawed feet.

Note that four default rules are required to represent the relationship between a gen and a specialization of the gen. For example, that G_3 differs from G_1 in eating fish added the following defaults:

1. A feature default that asserts that G_3's possess that feature (eating fish).
2. A link default that asserts that G_3's are differentiated from G_1's by that diff (eating fish).
3. A differentiated norm default that makes explicit the exception that G_1's usually eat insects, unless they eat fish.
4. A restricted closed world default that makes explicit the assumption that G_1's typically do not eat fish.

The case memory is used to reason in the following manner. We always assume that an individual is an instance of the root of the memory. In addition, several features about the individual are asserted. We use these features to traverse the memory in order to find generalizations that match the given individual. Simultaneously, we collect norms from the gens encountered along each path following the algorithm presented in the previous section. In this way, we are able to infer that the individual possesses features that were not explicitly given.

In the context of the corresponding default theory, we are given some first-order assertions and then construct an extension. The extension will consist of a consistent set of beliefs that can be concluded from the theory and these assertions.

For example, if we are told that *cover(Tweety, feathers)*, *feet(Tweety, claw)*, and *habitat(Tweety, water)* the following extension is constructed:

$$\left\{ \begin{array}{ll} G_3(Tweety), & G_2(Tweety), \\ G_1(Tweety), & G_0(Tweety), \\ diet(Tweety, fish), & habitat(Tweety, water), \\ feet(Tweety, claw), & cover(Tweety, feathers) \end{array} \right\},$$

namely that Tweety is an instance of the generalizations G_3, G_2, G_1, and G_0; and that Tweety's diet consists of fish, Tweety's habitat is water, Tweety's feet are clawed, and Tweety is covered with feathers.

Note that the extension is unique because the pairs of DN / RCW defaults allow features from more specific gens to override the norms from more general ones.[4]

For example, the DN default

$$\frac{G_2(x) \wedge \neg feet(x, claw) : feet(x, web)}{feet(x, web)}$$

in conjunction with the the explicitly given feature *feet(Tweety, claw)* blocked inheriting $feet(x, web)$ from G_2, while the DN default

[4] These pairs of defaults are similar to the way Reiter and Criscuolo handle interactions between defaults that are usually translated into semi-normal defaults.[Reiter and Criscuolo, 1983]

$$W = \{\forall x.G_0(x)\}$$

$$D = \left\{ \begin{array}{ll}
\frac{G_0(x):cover(x,feathers)}{cover(x,feathers)}, & \frac{G_0(x) \wedge feet(x,claw):G_1(x)}{G_1(x)}, \\
\frac{G_0(x) \wedge feet(x,web):G_2(x)}{G_2(x)}, & \frac{G_0(x) \wedge habitat(x,land):G_1(x)}{G_1(x)}, \\
\frac{G_0(x) \wedge habitat(x,water):G_2(x)}{G_2(x)}, & \frac{G_0(x) \wedge diet(x,insect):G_1(x)}{G_1(x)}, \\
\frac{G_0(x) \wedge diet(x,fish):G_2(x)}{G_2(x)}, & \frac{G_1(x) \wedge habitat(x,water):G_3(x)}{G_3(x)}, \\
\frac{G_1(x) \wedge diet(x,insect):G_3(x)}{G_3(x)}, & \frac{G_2(x) \wedge feet(x,claw):G_3(x)}{G_3(x)}, \\
\frac{G_1(x) \wedge \neg habitat(x,water):habitat(x,land)}{habitat(x,land)}, & \frac{G_1(x) \wedge \neg diet(x,fish):diet(x,insect)}{diet(x,insect)}, \\
\frac{G_1(x) \wedge \neg diet(x,fish):\neg habitat(x,water)}{\neg habitat(x,water)}, & \frac{G_1(x) \wedge \neg habitat(x,water):\neg diet(x,fish)}{\neg diet(x,fish)}, \\
\frac{G_2(x) \wedge \neg feet(x,claw):feet(x,web)}{feet(x,web)}, & \frac{G_2(x):\neg feet(x,claw)}{\neg feet(x,claw)}, \\
\frac{G_3(x):habitat(x,water)}{habitat(x,water)}, & \frac{G_3(x):diet(x,fish)}{diet(x,fish)}, \\
\frac{G_3(x):feet(x,claw)}{feet(x,claw)} &
\end{array} \right\}$$

Figure 2: Default Theory for Memory Fragment.

$$\frac{G_1(x) \wedge \neg habitat(x, water) : habitat(x, land)}{habitat(x, land)}$$

along with the given feature *habitat(Tweety, water)* blocked inheriting $habitat(x, land)$ G_1. The explicit feature, *habitat(x, water)*, combined with the RCW default

$$\frac{G_1 \wedge \neg habitat(x, water) : \neg diet(x, fish)}{\neg diet(x, fish)}$$

and the DN default

$$\frac{G_1(x) \wedge \neg diet(x, fish) : diet(x, insect)}{diet(x, insect)}$$

blocks inheriting $diet(x, insect)$ from G_1.

Thus, the default theory permitted us to generate the taxonomic relations implicit in the case memory and correctly infer default features through inheritance.

On the other hand, we may have genuinely ambiguous memory structure that give rise to multiple extensions. The familiar Quaker-Republican example is depicted as a fragment of case memory in Figure 3. We can imagine this memory structure as having been built up by encountering a number of pacifist Quakers and non-pacifist Republicans. The corresponding default theory shown in Figure 4.

If all we are told is $religion(Dick, quaker)$ and $party(Dick, republican)$, we will generate the the following two extensions, depending upon the order in which the defaults are applied:

$$E_1 = \left\{ \begin{array}{l} G_1(Dick), \\ G_0(Dick), \\ religion(Dick, quaker), \\ party(Dick, republican), \\ pacifist(Dick, yes) \end{array} \right\}$$

and

$$E_2 = \left\{ \begin{array}{l} G_2(Dick), \\ G_0(Dick), \\ party(Dick, republican), \\ religion(Dick, quaker), \\ pacifist(Dick, no) \end{array} \right\}.$$

5 Discussion

The ability to represent the kinds of knowledge present in a case memory structure as a collection of normal defaults is reassuring. Normal default theories possess several nice properties. In particular:

1. normal default theories are coherent, i.e., they possess at least one extension, and
2. normal default theories are semi-monotonic, i.e., if D_1 and D_2 are sets of normal default rules and E_1 is an extension for (D_1, W), then there is an extension of the theory $(D_1 \cup D_2)$, E_2, such that $E_1 \subseteq E_2$.

The first property assures us that we can derive a consistent of features possessed by an individual. The second property assures us that when the changes made to a case memory only require adding defaults to the corresponding default theory, there is a consistent set of features that can be derived in the new memory structure that contains a consistent set of features derived in the old one. In general, however, the changes to the case memory correspond to deleting, as well as adding, defaults in the corresponding default theory, so we are not guaranteed of semi-monotonicity. The non-monotonicity of case memory should not be surprising, since the self-structuring process involves creating new generalizations as well as creating new specializations.

The Quaker-Republican example reveals that, as expected, the default theory corresponding to a case memory may give rise to multiple extensions. Indeed, in a complex memory structure multiple extensions may be the rule. Typically, case-based reasoning systems use extra-logical machinery to prefer one collection of features over another.

The algorithm presented in Section 3 will find all possible sets of features along each path. When there are multiple extensions in the underlying default theory, some of these sets may contain incompatible features. We believe that each set of features, which corresponds to traversing a single path from the root to a node in the memory, lies within a single extension of the corresponding default theory, at least for "naturally" occurring case memories. Traversing a single path essentially serves to choose an order in which to apply the defaults, and because the path goes from a gen to another node specialized from the gen by possessing a diff, a consistent collection of defaults are applied.

It might appear surprising that the correspondence between case memory and default logic made use of normal defaults alone, while inheritance networks require semi-normal defaults[Reiter, 1980, Etherington, 1988]. The reason is that the case memory is a discrimination network and the links between nodes represent specializing a more general node to a more specific node. Likewise, it is reasonable, without any further knowledge, to assume that if differences are not made explicit that they should be assumed not to hold. Thus we can use pairs of normal defaults to block unwanted conclusions instead of using semi-normal defaults to represent explicit exceptions.

Acknowledgments: Janet Kolodner's work on self-organizing memory structures laid the foundation for the ideas presented in this paper.

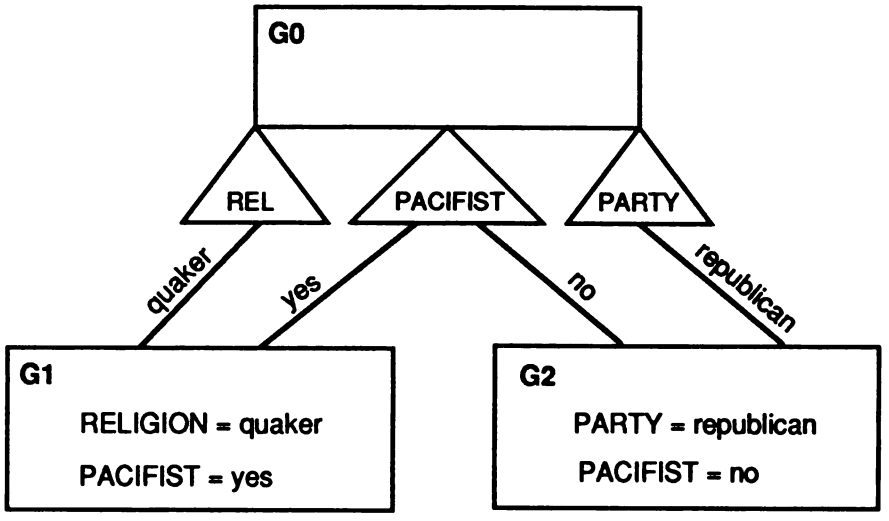


Figure 3: The Quaker-Republican Example.

$$W = \left\{ \begin{array}{l} \forall x.G_0(x), \\ \forall x.pacifist(x,yes) \supset \neg pacifist(x,no), \\ \forall x.pacifist(x,no) \supset \neg pacifist(x,yes) \end{array} \right\}$$

$$D = \left\{ \begin{array}{ll} \dfrac{G_0(x) \wedge religion(x,quaker) : G_1(x)}{G_1(x)}, & \dfrac{G_0(x) \wedge party(x,republican) : G_2(x)}{G_2(x)}, \\ \dfrac{G_0(x) \wedge pacifist(x,yes) : G_1(x)}{G_1(x)}, & \dfrac{G_0(x) \wedge pacifist(x,no) : G_2(x)}{G_2(x)}, \\ \dfrac{G_1(x) : pacifist(x,yes)}{pacifist(x,yes)}, & \dfrac{G_1(x) : religion(x,quaker)}{religion(x,quaker)}, \\ \dfrac{G_2(x) : party(x,republican)}{party(x,republican)}, & \dfrac{G_2(x) : pacifist(x,no)}{pacifist(x,no)} \end{array} \right\}$$

Figure 4: The Quaker-Republican Default Theory.

References

[Etherington, 1987] David W Etherington. More on inheritance hierarchies with exceptions. In *Proceedings of the National Conference on Artificial Intelligence*, pages 352–357, American Association for Artificial Intelligence, 1987.

[Etherington, 1988] David W Etherington. *Reasoning with Incomplete Information.* Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1988.

[Etherington and Reiter, 1985] David W. Etherington and Raymond Reiter. On inheritance hierarchies with exceptions. In Ronald J. Brachman and Hector J. Levesque, editors, *Readings in Knowledge Representation*, chapter 17, pages 330–334, Morgan Kaufmann Publishers, Inc., Los Altos, California, 1985.

[Kolodner, 1983] Janet L. Kolodner. Maintaining organization in a dymanic long-term memory. *Cognitive Science*, 7:243–280, 1983.

[Kolodner, 1984] Janet L. Kolodner. *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.* Lawrence Erlbaum Associates, Inc., Hillside, NJ, 1984.

[Reiter, 1980] Raymond Reiter. A logic for default reasoning. In Matthew L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*, pages 68–93, Morgan Kaufmann Publishers, Inc., Los Altos, California, 1987.

[Reiter and Criscuolo, 1983] Raymond Reiter and Giovanni Criscuolo. Some representational issues in default reasoning. *Computers and Mathematics with Applications*, 9:15–27, 1983.

What does a conditional knowledge base entail?

Daniel Lehmann*
Department of Computer Science,
Hebrew University, Jerusalem 91904 Israel

Abstract

This paper presents a purely logical approach to nonmonotonic reasoning. A conditional knowledge base, consisting of a set of conditional assertions of the type **if** ... **then** ..., represents the explicit defeasible knowledge an agent has about the way the world generally behaves. The set of all conditional assertions entailed by a conditional knowledge base is studied. This set does not grow monotonically with the knowledge base. This paper relies on both a proof-theoretic approach based on cumulative logic, as proposed by D. Gabbay in [Gabbay, 1985], and a semantical approach, as advocated by Y. Shoham in [Shoham, 1988] and [Shoham, 1987]. Two families of consequence relations are central to this work: preferential and rational consequence relations. Those families have been characterized in [Kraus *et al.*,] and [Lehmann and Magidor,] as those consequence relations defined respectively by two families of models. Those models are sets of possible states equipped with a preference relation, some states being preferable, i.e. more natural, than others.

1 Introduction and background

This paper surveys and extends previous results obtained jointly with Sarit Kraus and Menachem Magidor. Preliminary versions of those results may be found in [Lehmann and Kraus, 1988], [Lehmann, 1988], [Kraus *et al.*, 1988a] and [Lehmann and Magidor, 1988].

Inference is the process of achieving explicit information that was only implicit in the agent's knowledge. Human beings are astoundingly good at infering useful and very often reliable information from knowledge that seems mostly irrelevant, sometimes erroneous and even self-contradictory. They are even better at correcting inferences they learn to be in contradiction with reality. It is already a decade now that AI has realized that the analysis of such inference processes was a major task.

Many examples of nonmonotonic systems have been proposed as formal models of this kind of inferences. The best known are probably: circumscription [McCarthy, 1980], the modal system of [McDermott and Doyle, 1980], default logic [Reiter, 1980] and negation as failure [Clark, 1978]. An up-to-date survey of the field of nonmonotonic reasoning may be found in [Reiter, 1987].

Though each of these systems is interesting per se, it is not clear that any one of them really captures the whole generality of nonmonotonic reasoning. Recently (see in particular the panel discussion of [Vardi, 1988]) a number of researchers expressed their disappointment at existing systems and suggested that no purely logical analysis could be satisfactory.

This work tries to contradict this pessimistic outlook. It takes a purely logical approach, grounded in A. Tarski's framework of consequence relations [Tarski, 1956] and studies the very general notion of a *sensible* conclusion. It seems that this a common ground that can be widely accepted: all reasonable inference systems draw only sensible conclusions. On the other hand, as will be shown, the notion of a sensible conclusion has a non-trivial mathematical theory and many interesting properties are shared by all ways of drawing sensible conclusions.

D. Gabbay [Gabbay, 1985] was probably the first to suggest to focus the study of nonmonotonic logics on their consequence relations. Y. Shoham in [Shoham, 1988] and [Shoham, 1987] proposed a general model theory for nonmonotonic inference. He suggested models that may be described as a set of worlds equipped with a *preference* relation: the preference relation is a partial order (sometimes supposed to be well-founded) and a world v is preferable to some other world w if v is

*The elaboration of this work was partially supported by the Jean and Helene Alfassa fund for research in Artificial Intelligence.

more *normal* than w. Shoham claimed that adequate semantics could be given to all known nonmonotonic systems by using such a preference relation.

This paper shows that the proof-theoretic and the model-theoretic approaches are closely related. The clarification of the relations between proofs and models that has been achieved and is surveyed here allows for the (future) design of efficient decision procedures tuned to different restrictions on the language of propositions or the knowledge bases. Such decision procedures (or heuristics) could be the core of automated engines of sensible inferences.

This paper is intended to be a short readable introduction to a growing body of work, aimed at readers already acquainted with the field of nonmonotonic reasoning. Therefore it is short on motivation and concentrates on explaining the meaning of technical results and neither provides proofs nor tries to present the most general possible results. Two assumptions must be mentioned. The underlying set of formulas is assumed to be a propositional language. If one wants to express formulas in first order predicate calculus, some additional rules of inference have to be introduced to deal with the quantifiers. But the main results and the construction of the logical closure of a conditional knowledge base are unchanged. The distinction between *soft* i.e. *defeasible* and *hard* constraints is not dealt with here. It is assumed that all pieces of information, i.e. conditional assertions, are soft constraints. Dealing with hard constraints, in addition to soft ones, involves relativizing to some given set of tautologies and poses no problem at all.

In section 11 the relations of this work and some other different lines of enquiry are described. Since it contains some technical points relating to the material presented here, it has been left for the end of the paper. In section 2 the syntax is defined. In section 3 a number of inference rules are defined and discussed. The rules provide the definition of two families of consequence relations: preferential and rational consequence relations. The discussion helps grasping the intended meaning of conditional assertions. In section 4 preferential models are introduced and shown to define exactly the preferential consequence relations. In section 5 ranked models are introduced and shown to define exactly the rational consequence relations. In section 6 the notion of preferential entailment is characterized and it is shown that the problem of computing preferential entailment has low complexity: it is in co-NP. The question put forward in the title is discussed in section 7 and further studied in section 8. Section 9 discusses a standard example and section 10 tries to charter the course of future research.

2 The language

The first step is to define a language in which to express the basic propositions. In this paper Propositional Calculus is chosen. Let L be the set of well formed propositional formulas (thereafter formulas) over a set of propositional variables. The classical propositional connectives will be denoted by $\neg, \vee, \wedge, \rightarrow$ and $\leftrightarrow$. The connective $\rightarrow$ therefore denotes material implication. Small greek letters will be used to denote formulas.

A world is an assignment of truth values to the propositional variables. The set $\mathcal{U}$ is the set of all worlds. The satisfaction of a formula by a world is defined as usual. The notions of satisfaction of a set of formulas, validity of a formula and satisfiability of a set of formulas are defined as usual. We shall write $\models \alpha$ if α is valid, i.e. iff $\forall u \in \mathcal{U}$, $u \models \alpha$.

If α and β are formulas then the pair $\alpha \mid\!\sim \beta$ (read from α sensibly conclude β) is called a conditional assertion. A conditional assertion is a syntactic object to which the reader may attach any meaning he wants, but the meaning we attach to such an assertion, and against which the reader should check the logical systems to be presented in the upcoming sections, is the following: if α represents the information I have about the true state of the world, I will jump to the conclusion that β is true.

A conditional knowledge base is any set of conditional assertions. Typically it is a finite set, but need not be so. Conditional knowledge bases seem to provide a terse and versatile way of specifying defeasible information. They correspond to the explicit information an agent may have.

Certain well-behaved sets of conditional assertions will be deemed worthy of being called *consequence relations*. We shall use the notation usual for binary relations to describe consequence relations. So, if $\mid\!\sim$ is a consequence relation, $\alpha \mid\!\sim \beta$ indicates that the pair $\langle \alpha, \beta \rangle$ is in the consequence relation $\mid\!\sim$ and $\alpha \not\mid\!\sim \beta$ indicates it is not in the relation. Consequence relations correspond to the implicit information an intelligent agent may have. Consequence relations are typically infinite sets.

3 Rules

3.1 Preferential relations

Certain especially interesting properties of sets of conditional assertions (i.e. binary relations on L) will be described and discussed now. They are presented in the form of inference rules. Consequence relations are expected to satisfy those properties.

$$\frac{\models \alpha \leftrightarrow \beta \ , \ \alpha \vdash \gamma}{\beta \vdash \gamma} \quad \textbf{(Left Logical Equivalence)} \quad (1)$$

$$\frac{\models \alpha \rightarrow \beta \ , \ \gamma \vdash \alpha}{\gamma \vdash \beta} \quad \textbf{(Right Weakening)} \quad (2)$$

$$\alpha \vdash \alpha \quad \textbf{(Reflexivity)} \quad (3)$$

$$\frac{\alpha \vdash \beta \ , \ \alpha \vdash \gamma}{\alpha \vdash \beta \wedge \gamma} \quad \textbf{(And)} \quad (4)$$

$$\frac{\alpha \vdash \gamma \ , \ \beta \vdash \gamma}{\alpha \vee \beta \vdash \gamma} \quad \textbf{(Or)} \quad (5)$$

$$\frac{\alpha \vdash \beta \ , \ \alpha \vdash \gamma}{\alpha \wedge \beta \vdash \gamma} \quad \textbf{(Cautious Monotony)} \quad (6)$$

Definition 1 *A set of conditional assertions that satisfies all six properties above is called a* preferential *consequence relation.*

Considered as inference rules the six properties above constitute a logical system that we shall denote by **P**. The system **P** is essentially the flat (i.e. unnested) part of the system **SS** of J. Pollock. A larger family of consequence relations: cumulative relations, has been studied in [Kraus *et al.*,]. This family is closely related to the cumulative inference operations studied by D. Makinson in [Makinson, 1989].

Before we mention some derived rules, let us justify the rules above. The rules of **Right Weakening** and **Reflexivity** seem to be valid universally in any kind of reasoning. The rule of **Left Logical Equivalence**, that the conclusions one can draw from a proposition α depend only on the meaning of α, not on its form, also seems unavoidable. **Cautious Monotony** says that learning of a new fact, the truth of which was expected cannot invalidate a previous conclusion. This is, for us, and following D. Gabbay [Gabbay, 1985], the central property of the inference systems of interest to us. This condition is named *triangulation* in [Pearl and Geffner, 1988] and is similar to axiom **A3** of [Burgess, 1981]. Some probabilistic interpretations, though, e.g. interpreting a conditional assertion as meaning that the corresponding conditional probability is larger than some $p < 1$, invalidate **Cautious Monotony**. It is valid in Adams' system [Adams, 1975]. Similarly, the rules **And** does not seem objectionable, if one rejects the semantics described just above. The rule **Or** is more disputable. First, as **Cautious Monotony** and **And**, it does not fit the simple minded probabilistic semantics just described. But, there is also an *epistemic* reading of $\alpha \vdash \beta$ that invalidates the **Or** rule. If we interpret $\alpha \vdash \beta$ as meaning: *if all I know about the world is α then it is sensible for me to suppose that β is true*, we must reject the **Or** rule. Indeed, one may imagine a situation in which α expresses a fact that can very well be true or false but the truth value of which is normally not known to me. If I knew α to be true, that would be a quite abnormal situation in which I may be willing to accept γ. If I knew α to be false, similarly, it would be an exceptional situation in which I may accept γ, but the knowledge that $\alpha \vee \neg\alpha$ is true is essentially void and certainly does not allow me to conclude that anything exceptional is happening. Notice that, in this reading, the left hand side of the symbol $\vdash$ involves a hidden epistemic operator (the right hand side may also do so, but need not). It is therefore possible to defend the **Or** rule by saying that such a hidden operator should be made explicit and the example just above only invalidates the inference: from $\mathcal{K}\alpha \vdash \gamma$ and $\mathcal{K}\beta \vdash \gamma$ infer $\mathcal{K}(\alpha \vee \beta) \vdash \gamma$. But nobody would defend such an inference anyway.

For the reader's ease of mind we shall mention two important derived rules. Both **S** and **Cut** are satisfied by any preferential relation.

$$\frac{\alpha \wedge \beta \vdash \gamma}{\alpha \vdash \beta \rightarrow \gamma} \quad \textbf{(S)} \quad (7)$$

$$\frac{\alpha \wedge \beta \vdash \gamma \ , \ \alpha \vdash \beta}{\alpha \vdash \gamma} \quad \textbf{(Cut)} \quad (8)$$

3.2 Rational relations

A more restricted family of consequence relations will be defined now.

Definition 2 *A preferential consequence relation $\vdash$ is said to be* rational *iff it satisfies the following condition of* **Rational Monotony**: *if $\alpha \vdash \gamma$ and $\alpha \wedge \beta \not\vdash \gamma$ then $\alpha \vdash \neg\beta$.*

Rational Monotony is similar to **CV** of conditional logic. From the results of sections 4 and 5, the reader will easily see that there are preferential relations that are not rational. The justification for **Rational Monotony** is the following: if I am ready to conclude (sensibly) that γ holds because α is true but am not ready any more to conclude γ if I learn that, in addition to α, β is also true, then it must be the case that, learning that β is true in addition to α is surprising, i.e., on the basis of α, I expected β to be false. **Rational Monotony** expresses some kind of restricted monotony. The fact that it is not satisfied by all preferential consequence relations means that some of those relations lack a degree of monotony that seems natural.

The following are properties of rational relations that are not enjoyed by all preferential relations.

Lemma 1 *Let $\vdash$ be a rational relation. If $\alpha \vee \beta \vdash \gamma$ then either $\alpha \vdash \gamma$ or $\beta \vdash \gamma$ (or both). If $\alpha \vdash \gamma$ then either $\alpha \wedge \beta \vdash \gamma$ or $\alpha \wedge \neg\beta \vdash \gamma$ (or both).*

Lemma 1 implies that rational consequence relations represent ways of infering information in which one never draws conclusions solely on the basis of ignorance. Its failure for preferential relations indicates that not all preferential consequence relations represent bona fide nonmonotonic reasoning.

4 Preferential models

In this section, preferential models are defined and shown to fit exactly the preferential consequence relations. This is an important step in the study of preferential and rational relations. Those models are called *preferential* because they represent a slight variation on those proposed in [Shoham, 1987]. The differences are nevertheless technically important.

Preferential models give a model-theoretic account of the way one performs nonmonotonic inferences. The main idea is that the agent has, in his mind, a partial ordering on possible states of the world. State s is less than state t, if, in the agent's mind, s is *preferred* to or more *natural* than t. Now, the agent is willing to conclude β from α, if all *most natural* states that satisfy α also satisfy β.

Some technical definitions are needed. Let U be a set and $\prec$ a strict partial order on U, i.e. a binary relation that is antireflexive and transitive.

Definition 3 *Let $V \subseteq U$. We shall say that $t \in V$ is minimal in V iff there is no $s \in V$, such that $s \prec t$.*

Definition 4 *Let $V \subseteq U$. We shall say that V is smooth iff $\forall t \in V$, either $\exists s$ minimal in V, such that $s \prec t$ or t is itself minimal in V.*

We may now define the family of models we are interested in.

Definition 5 *A* preferential *model W is a triple $\langle S, l, \prec \rangle$ where S is a set, the elements of which will be called states, $l : S \mapsto \mathcal{U}$ assigns a world to each state and $\prec$ is a strict partial order on S satisfying the following* smoothness condition*: $\forall \alpha \in L$, the set of states $\widehat{\alpha} \stackrel{\text{def}}{=} \{s \mid s \in S, s \not\models \alpha\}$ is smooth, where $\not\models$ is defined as $s \not\models \alpha$ (read s satisfies α) iff $l(s) \models \alpha$.*

The smoothness condition is only a technical condition needed to deal with infinite sets of formulas, it is always satisfied in any preferential model in which S is finite, or in which $\prec$ is well-founded (i.e. no infinite descending chains). The requirement that the relation $\prec$ be a strict partial order has been introduced only because such models are nicer and the smoothness condition is easier to check on those models, but the soundness result is true for the larger family of models, where $\prec$ is just any binary relation (definitions 3 and 4 may make sense even when $\prec$ is any binary relation). In such a case, obviously, the smoothness condition cannot be dropped even for finite models. The completeness result holds obviously too for the larger family, but is less interesting.

We shall now describe the consequence relation defined by a model.

Definition 6 *Suppose a model $W = \langle S, l, \prec \rangle$ and $\alpha, \beta \in L$ are given. The consequence relation defined by W will be denoted by $\vdash_W$ and is defined by: $\alpha \vdash_W \beta$ iff for any s minimal in $\widehat{\alpha}$, $s \not\models \beta$.*

If $\alpha \vdash_W \beta$ we shall say that the model W satisfies the conditional assertion $\alpha \vdash \beta$, or that W is a model of $\alpha \vdash \beta$.

The following characterizes preferential consequence relations.

Theorem 1 (Kraus, Lehmann and Magidor) *A binary relation $\vdash$ on L is a preferential consequence relation iff it is the consequence relation defined by some preferential model.*

5 Ranked models

We may now characterize in a similar way the rational consequence relations. Ranked models are a subfamily of preferential models in which the partial order $\prec$ is in a way well-behaved. They provide an appealing model-theoretic account of nonmonotonic reasoning.

Definition 7 *A ranked model W is a preferential model $\langle S, l, \prec \rangle$ for which the strict partial order $\prec$ may be defined in the following way: there is a totally ordered set Ω (the strict order on Ω will be denoted by $<$) and a function $r : S \mapsto \Omega$ such that $s \prec t$ iff $r(s) < r(t)$.*

Those models are said to be ranked since the effect of function r is to rank the states: a state of smaller rank being more normal than a state of higher rank. Notice that we still require W to satisfy the smoothness condition. It is easy to see that for any subset T of S and any $t \in T$, t is minimal in T iff $r(t)$ is the minimum of the set $r(T)$. It follows that all minimal elements of T have the same image by r. The smoothness condition is then equivalent to the following: for any formula $\alpha \in L$, if $\widehat{\alpha}$ is not empty, the set $r(\widehat{\alpha})$ has a minimum ($\widehat{\alpha}$ has been defined to be the set of all states of S that satisfy α). The smoothness condition is always verified if Ω is a well-ordered set. It was shown in [Lehmann and Magidor,] that the characterization of theorem 2 would not be correct, had we dropped the requirement that Ω be well-ordered.

The rational consequence relations are characterized by the following:

Theorem 2 (Lehmann and Magidor) *A consequence relation is rational iff it is defined by some ranked model.*

6 Preferential entailment

6.1 Definition

It is time to begin to try to answer the question of the title. Let **K** be a set of conditional assertions and $\mathcal{A}$ some conditional assertion. Since it has been argued above that any bona fide nonmonotonic reasoning system should define a rational consequence relation and in view of theorem 2, the natural answer to the question of the title seems to be the following: the conditional assertion $\mathcal{A}$ is entailed by the set of conditional assertions **K** iff it is satisfied by all *ranked* models that satisfy all the assertions of **K**. Unfortunately this first try cannot provide a satisfactory answer, as will be seen now.

Theorem 3 *If the assertion $\mathcal{A}$ is satisfied by all* ranked *models that satisfy all the assertions of* **K** *then it is satisfied by all* preferential *such models.*

Theorem 3 is a precise way of describing the problems caused by the first temporary definition of section 4 of [Delgrande, 1988], and noticed there.

Before we can assess the implications of theorem 3, and explain why it is disappointing, let us lay down a definition and some results.

Definition 8 *The assertion $\mathcal{A}$ is* preferentially entailed *by* **K** *iff it is satisfied by all preferential models of* **K**. *The set of all conditional assertions that are preferentially entailed by* **K** *will be denoted by* $\mathbf{K}^p$. *The preferential consequence relation* $\mathbf{K}^p$ *is called the preferential closure of* **K**.

The characterization of preferential consequence relations obtained in theorem 1 enables us to prove the following.

Theorem 4 *Let* **K** *be a set of conditional assertions, and $\mathcal{A}$ a conditional assertion. The following conditions are equivalent:*

1. *$\mathcal{A}$ is preferentially entailed by* **K**, *i.e.* $\mathcal{A} \in \mathbf{K}^p$
2. *$\mathcal{A}$ has a proof from* **K** *in the system* **P**.

The following compactness result follows.

Corollary 1 (compactness) **K** *preferentially entails $\mathcal{A}$ iff a finite subset of* **K** *does.*

The following also follows from theorem 4.

Corollary 2 *The set* $\mathbf{K}^p$, *considered as a consequence relation, is a preferential consequence relation, therefore there is a preferential model that satisfies exactly the assertions of* $\mathbf{K}^p$. *If* **K** *is itself a preferential consequence relation then* $\mathbf{K} = \mathbf{K}^p$. *The set* $\mathbf{K}^p$ *grows monotonically with* **K**.

6.2 Probabilistic entailment

In [Adams, 1975] E. Adams defines a notion of *probabilistic entailment*. Roughly speaking a conditional assertion $\mathcal{A}$ is probabilistically entailed by a set **K** of conditional assertions iff for all (suitable) probability assignments the conditional probability assigned to $\mathcal{A}$ can be made as close as one wants to one, if only one makes sure that the conditional probabilities of the elements of **K** are close enough to one. He showed that, *for finite sets* **K**, a conditional assertion $\mathcal{A}$ is probabilistically entailed by **K** iff it has a proof from **K** in the system **P**. For finite knowledge bases then, probabilistic entailment is preferential entailment. The result does not hold for infinite knowledge bases, though, since, as Adams remarked, probabilistic entailment does not satisfy the compactness property.

6.3 Discussion of preferential entailment

Corollary 2 explains why the notion of preferential entailment cannot be the one we are looking for: the relation $\mathbf{K}^p$ can be any preferential relation and is not in general rational. For typical **K**'s, $\mathbf{K}^p$ fails to satisfy a large number of instances of **Rational Monotony** and is therefore highly unsuitable. One particularly annoying instance of this is the following. Suppose a conditional knowledge base **K** contains one single assertion $p \mathrel{|\!\sim} q$ where p and q are different propositional variables. Let r be a propositional variable, different from p and q. We intuitively expect the assertion $p \wedge r \mathrel{|\!\sim} q$ to follow from **K**. The rationale for that has been discussed extensively in the literature and boils down to: since we have no information whatsoever about the influence of r on objects satisfying p it is sensible to assume that it has not and that there are normal p-objects that satisfy r; the normal $p \wedge r$-objects are therefore normal p-objects and have all the properties enjoyed by normal p-objects. Nevertheless it is easy to check that $p \wedge r \mathrel{|\!\sim} q$ is not in $\mathbf{K}^p$. The problem lies, at least in part, with the fact that $\mathbf{K}^p$ is not rational, since any rational relation containing $p \mathrel{|\!\sim} q$, must contain $p \wedge r \mathrel{|\!\sim} q$ unless it contains $p \mathrel{|\!\sim} \neg r$. This question will be brought up again in section 7.2 and a solution will be proposed.

6.4 Complexity of preferential entailment

Nevertheless preferential entailment is a central concept and it is therefore worthwhile to study its computational complexity. The results here are quite encouraging: the problem is in co-NP, i.e. in the same polynomial class as the problem of deciding whether a propositional formula is valid.

Theorem 5 *There is a non-deterministic algorithm that, given a finite set* **K** *of conditional assertions and a conditional assertion* $\mathcal{A}$*, checks that* $\mathcal{A}$ *is not preferentially entailed by* **K**. *The running time of this algorithm is polynomial in the size of* **K** *(sum of the sizes of its elements) and* $\mathcal{A}$.

Proof: Let **K** be $\{\gamma_i \vdash \delta_i\}_{i=1}^N$. Let $I \subseteq \{1,\ldots,N\}$ be a set of indexes. We shall define: $\phi_I \stackrel{\text{def}}{=} \bigvee_{i\in I} \gamma_i$ and $\psi_I \stackrel{\text{def}}{=} \bigwedge_{i\in I} (\gamma_i \rightarrow \delta_i)$. A sequence is a sequence of pairs (I_i, f_i) for $i = 0,\ldots,n$, where $I_i \subseteq I$ and f_i is a world. Let α and β be in L.

Definition 9 *A sequence* (I_i, f_i), $i = 0,\ldots,n$, *is a witness for* $\alpha \vdash \beta$ *(we mean a witness that* $\alpha \vdash \beta$ *is not preferentially entailed by* **K** *) iff*

1. $f_k \models \psi_{I_k}$, $\forall k = 0,\ldots,n$
2. $f_k \models \phi_{I_k}$, $\forall k = 0,\ldots,n-1$
3. $I_{k+1} = I_k \bigcap \{j \mid f_k \not\models \gamma_j\}$, $\forall k = 0,\ldots,n-1$
4. $I_0 = \{1,\ldots,N\}$
5. $f_k \not\models \alpha$, $\forall k = 0,\ldots,n-1$
6. $f_n \models \alpha \wedge \neg\beta$.

We must check that: witnesses are short and a conditional assertion has a witness iff it is not preferentially entailed by **K**.

For the first point, just remark that, for $k = 0,\ldots,n-1$ the inclusion $I_k \supset I_{k+1}$ is strict because of items 3 and 2. The length of the sequence is therefore bounded by the number of assertions in **K**. Each pair, on the other hand has a short description.

For the second point, suppose first there is a witness for $\alpha \vdash \beta$. Then the ranked model W consisting of worlds $f_0,\ldots,f_n$ where $f_k \prec f_{k+1}$ $\forall k = 0,\ldots,n-1$ satisfies **K** but not $\alpha \vdash \beta$. That it does not satisfy $\alpha \vdash \beta$ is clear from items 5 and 6. Let us check that W satisfies $\gamma_i \vdash \delta_i$. If none of the f_k's, $k = 0,\ldots,n$ satisfies γ_i then W satisfies $\gamma_i \vdash \eta$ for any η in L. Suppose therefore that j is the smallest k for which $f_j \models \gamma_i$. We must show that $f_j \models \delta_i$. But, by items 4 and 3 $i \in I_j$ and by item 1, $f_i \models \delta_i$.

Suppose now that $\alpha \vdash \beta$ is not preferentially entailed by some given finite **K**. By techniques used in the proof of theorem 3 one may show that there is a finite linearly ordered model W of **K**, no state of which satisfies α, except the top state that is labeled by a world m that satisfies $\alpha \wedge \neg\beta$. Let $I_0 \stackrel{\text{def}}{=} \{1,\ldots,N\}$. It is easy to see that (*remark* 1): if V is any preferential model of **K**, for any set $I \subseteq I_0$, V satisfies $\phi_I \vdash \psi_I$. Let us first consider the set $\alpha \widehat{\vee \phi_{I_0}}$. It cannot be empty, therefore it has a unique minimal state. Let f_0 be the label of this state. We must consider two cases. First suppose that $f_0 \models \alpha$. Then f_0 is minimal in $\widehat{\alpha}$ and therefore must be m. In such a case (I_0, m) is a witness. The only thing to check is that item 1 is satisfied. Indeed either $m \models \phi_{I_0}$ and we conclude by remark 1 or $\widehat{\phi_{I_0}} = \emptyset$ and m satisfies none of the γ_i's. Let us deal now with the case $f_0 \not\models \alpha$. We shall build a sequence beginning by (I_0, m). Since m does not satisfy α, it must satisfy ϕ_{I_0}, which takes care of item 2. Remark 1 takes care of item 1. Let us now define $I_1 = I_0 \bigcap \{j \mid f_0 \not\models \gamma_j\}$. I_1 is strictly smaller than I_0. We may now consider the set $\alpha \widehat{\vee \phi_{I_1}}$. It is not empty and therefore has a unique minimal element and we may, in this way, go on and build a proof for $\alpha \vdash \beta$.

7 The rational closure of a conditional knowledge base

7.1 Perfect extensions

All has been done so far does not allow us to give a satisfactory answer to the question of the title. Let **K** be a set of conditional assertions. We would like to define a consequence relation $\overline{\mathbf{K}}$, the rational closure of **K**, that contains all the conditional assertions that we intuitively expect to follow from **K**.

At this point the reader should be convinced that $\overline{\mathbf{K}}$ must be a rational consequence relation that extends **K**. Any such relation also extends $\mathbf{K}^p$.

The techniques used in the proofs of the results presented in section 6.1 show that not only is there no unique rational extension of **K**, there is no minimal such rational extension. There is obviously a maximal such extension: the full consequence relation, (i.e. $\alpha \vdash \beta$ for all α, β in L) but this is certainly not the one we are looking for. Can we find out a number of properties that we want $\overline{\mathbf{K}}$ to possess in order to, at least, narrow the field of possibilities? The general underlying idea is that the assertions that are in $\overline{\mathbf{K}}$ but not in $\mathbf{K}^p$ should be assertions that have *support* in $\mathbf{K}^p$, i.e. assertions $\alpha \vdash \beta$ such that α is of the form $\gamma \wedge \delta$ for some γ such that $\gamma \vdash \beta$ is in $\mathbf{K}^p$. Let us encapsulate this idea in definitions.

Definition 10 *An assertion* $\alpha \vdash \beta$ *is said to be supported by (or in)* $\mathbf{K}^p$ *iff* α *is of the form* $\gamma \wedge \delta$ *for some* γ *such that* $\gamma \vdash \beta$ *is in* $\mathbf{K}^p$.

Definition 11 *A rational extension* $\mathbf{K}'$ *of* $\mathbf{K}$ *is called perfect iff every assertion of* $\mathbf{K}'$ *is supported by* $\mathbf{K}^p$.

We may present the following disappointing result.

Lemma 2 *There are conditional knowledge bases that have no rational perfect extension.*

Proof: Let L be the set of all propositional formulas built out of the set of four propositional variables: $\{a, b, c, d\}$. Let W be the preferential model with three states: $\{s, t, u\}$, in which $s \prec t$ (and this is the only pair in the relation $\prec$) and s satisfies only a, t satisfies only b and u satisfies only c and d. Let $\mathbf{K}$ be the set of all conditional assertions satisfied in W. We claim that $\mathbf{K}$ has no rational perfect extension. Notice, first, that W satisfies $a \vee b \vdash \neg b$. This assertion is therefore in $\mathbf{K}$. Any ranked model satisfying $a \vee b \vdash \neg b$ must satisfy at least one of the following two assertions: $a \vee c \vdash \neg c$ or $b \vee c \vdash \neg b$. Any rational extension of $\mathbf{K}$ must therefore contain one of the two assertions above. But $a \vee c \vdash \neg c$ has clearly no support in $\mathbf{K}^p$ and therefore any perfect rational extension of $\mathbf{K}$ must contain: $b \vee c \vdash \neg b$. But W satisfies $c \vdash d$ and any ranked model satisfying both $b \vee c \vdash \neg b$ and $c \vdash d$ must also satisfy $b \vee c \vdash d$. Any perfect rational extension of $\mathbf{K}$ must therefore contain this last formula but it clearly lacks support in $\mathbf{K}^p$. We conclude that $\mathbf{K}$ has no perfect rational extension. ∎

It is therefore reasonable to look for less than perfect extensions. Perfection restricted to two special kinds of formulas will now be defined.

Definition 12 *A rational extension* $\mathbf{K}'$ *of a conditional knowledge base* $\mathbf{K}$ *is said to be c-perfect iff every assertion of* $\mathbf{K}'$ *of the form* $\alpha \vdash$ **false** *is supported by* $\mathbf{K}^p$.

Definition 13 *A rational extension* $\mathbf{K}'$ *of a conditional knowledge base* $\mathbf{K}$ *is said to be t-perfect iff every assertion of* $\mathbf{K}'$ *of the form* **true** $\vdash \alpha$ *is supported by* $\mathbf{K}^p$.

A rational extension that is both c-perfect and t-perfect will be called ct-perfect. The following is easy to prove and will help understand the meaning of definitions 12 and 13.

Lemma 3 *An assertion of the form* $\alpha \vdash$ **false** *is supported by* $\mathbf{K}^p$ *iff it is in* $\mathbf{K}^p$. *An assertion of the form* **true** $\vdash \alpha$ *is supported by* $\mathbf{K}^p$ *iff it is in* $\mathbf{K}^p$.

7.2 Rational closure

The remainder of this paper is devoted to describing a way to build, for any *finite* conditional knowledge base $\mathbf{K}$, a ct-perfect rational extension of $\mathbf{K}$. The result of this construction will be denoted $\overline{\mathbf{K}}$. Some additional properties of $\overline{\mathbf{K}}$ will be described. At this point our long preparatory work will pay off since we may use a model-theoretic construction to describe $\overline{\mathbf{K}}$.

For the rest of this paper, let us suppose that $\mathbf{K}$ is a *finite* conditional knowledge base. Let P be any *finite* set of propositional variables that contains all the variables appearing in $\mathbf{K}$. How the choice of P affects the construction will be studied later. Let L be the set of all propositional formulas on P. The language L is logically finite, i.e. finite up to logical equivalence. From now on L will be fixed. The first step is to build a *finite* preferential model W (i.e. W has a finite number of states) that defines $\mathbf{K}^p$. The fact that this can be done follows from results of [Lehmann and Magidor,]. How the choice of a specific W may affect the construction will be studied later. Let W be the preferential model $\langle S, l, \prec \rangle$.

The second step is to massage W gently so as to make it a ranked model. More precisely we shall describe a partial order $\prec'$, that is an extension of $\prec$, such that $W' \stackrel{\text{def}}{=} \langle S, l, \prec' \rangle$ is a ranked model. The consequence relation $\overline{\mathbf{K}}$ is the relation defined by W'.

Before we proceed to describe $\prec'$, let us remark that any relation $\prec'$ that extends $\prec$ and makes W' a ranked model will enable us to define a rational extension of $\mathbf{K}$ that is c-perfect. The consequence relation defined by W' is rational since W is ranked, it extends the consequence relation defined by W because $\prec'$ extends $\prec$ (a state minimal in the new order must have been minimal in the old order) and it is c-perfect because, since the set of states and the labeling function stay the same, the sets of formulas of the form $\alpha \vdash$ **false** that are satisfied by W and W' respectively are the same.

The way chosen to define $\prec'$ is the following: let the states of W sink as low as they can, respecting $\prec$. More precisely: let us define the height of a state $s \in S$ as the length of the longest ascending (under $\prec$) chain of states whose top element is s. States that are minimal in S have height zero. States that are minimal in the set of states that are not of height zero have height one, and so on. The relation $\prec'$ will be defined in the following way: $s \prec' t$ iff $height(s) < height(t)$. It is clear that $\prec'$ is an extension of $\prec$ and that W' is a ranked model.

It is left for us now to check that $\overline{\mathbf{K}}$, the relation defined by W' is a t-perfect extension of $\mathbf{K}$. But this follows from the fact that all states that were minimal in S in the model W are of height zero and therefore still minimal in S in the model W'. We may now state:

Theorem 6 *For any finite set* $\mathbf{K}$ *of conditional assertions there exists a rational ct-perfect extension of*

$\mathbf{K}^P$.

The question now is whether the rational closure $\overline{\mathbf{K}}$ of $\mathbf{K}$ really represents the set of all conditional assertions we intuitively expect to be entailed by $\mathbf{K}$.

There is a preliminary point that must be clarified first. At two separate points in our construction some arbitrary choice was made: the set P of propositional variables may be any finite set large enough to contain all the variables appearing in $\mathbf{K}$ and the model W may be chosen as any finite preferential model that defines $\mathbf{K}^P$. Do those choices affect the construction of $\overline{\mathbf{K}}$?

Let us first deal with the last choice, that of W. The answer is that the construction may be affected by the choice of W but one may, without any harm, restrict oneself to a subfamily of preferential models, the *honest* models and all honest models W will define the same rational closure.

Definition 14 *Let W be the preferential model $\langle S, l, \prec \rangle$. We shall say that W is* honest *iff for every $s \in S$ there is a formula $\alpha \in L$ such that s is a minimal element of $\widehat{\alpha}$.*

It is clear that, given any preferential model, removing from it all the states that are not minimal in any $\widehat{\alpha}$, leaves a preferential model that is equivalent to the first one, i.e. defines exactly the same consequence relation. There is therefore no harm in restricting ourselves to honest models.

Lemma 4 *The construction of $\overline{\mathbf{K}}$ above is not affected by the choice of W as long as W is honest.*

Lemma 4 follows from a proof-theoretic characterization of $\overline{\mathbf{K}}$ that will be described now. This proof-theoretic characterization is important per se. It provides an alternative way of defining $\overline{\mathbf{K}}$.

Given a conditional knowledge base $\mathbf{K}$ (not necessarily finite), for every $i = 0, 1, 2, \ldots$ we shall define, by an inductive procedure, the degree of a formula. Let $\vdash_0$ be the preferential closure of $\mathbf{K}$.

The state zero of the induction is: the set of formulas of degree strictly less than zero is empty.

Suppose now that $i \geq 0$ and that the set of formulas of degree less than i has been defined. The set of formulas of degree i will be defined in the following way: formulas of degree i are those formulas α that are not of degree less than i and that satisfy: $\forall \beta \in L$ such that $\alpha \vee \beta \vdash_0 \neg\alpha$, β is of degree less than i.

In the procedure above not every formula gets a degree. Let us give the degree ∞ to formulas that are not assigned a degree.

Lemma 5 *Let α be any formula. If degree$(\alpha) < \infty$ then, in any honest preferential model defining $\mathbf{K}^P$ there is a state s such that height(s) = degree(α), s satisfies α and no state of height less than height(s) satisfy α. If degree$(\alpha) = \infty$ then $\alpha \vdash$ false is in $\mathbf{K}^P$.*

Lemma 6 *If W is a finite honest preferential model that defines the preferential closure of $\mathbf{K}$, then the inductive procedure described above terminates, i.e., the degrees given to all formulas that are given finite degrees are bounded, and $\overline{\mathbf{K}}$ is the consequence relation defined by: $\alpha \vdash \beta$ iff degree$(\alpha) <$ degree$(\alpha \wedge \neg\beta)$ or degree$(\alpha) = \infty$.*

The claim that the choice of W (as long as it is honest) follows, since the iterative procedure depends only on $\mathbf{K}^P$, not on W.

The iterative procedure described above, that may be applied also to infinite knowledge bases, most probably gives a clue as how to extend the rational closure operation to infinite knowledge bases.

What about the choice of the set of variables P?

For the rest of this section, let P' be the finite set of variables that appear in $\mathbf{K}$, let p be a propositional variable that is not in P', let $P \supseteq P' \cup \{p\}$, let α and β be propositional formulas over P' and let γ be a formula over P.

Lemma 7 *$\alpha \vdash \beta$ is in the rational closure of $\mathbf{K}$ computed over P iff it is in the rational closure of $\mathbf{K}$ computed over P'*

Lemma 8 *If $\alpha \vdash$ false is not in the preferential closure of $\mathbf{K}$ then $\alpha \vdash p$ is not in the rational closure of $\mathbf{K}$.*

The proof of lemma 8 proceeds by showing that the canonical model of $\mathbf{K}^P$ is stable under the automorphism changing p into $\neg p$.

Corollary 3 *If $\alpha \vdash \gamma$ is in the rational closure of $\mathbf{K}$ then so is $\alpha \wedge p \vdash \gamma$.*

Lemma 9 *If $\alpha \wedge p \vdash \beta$ is in the rational closure of $\mathbf{K}$ then so is $\alpha \vdash \beta$.*

The practical meaning of corollary 3 and lemma 9 is that the choice of the set of propositional variables does not really matter. It also says that rational closure behaves as we expect regarding propositions about which we have no information at all. The notion of rational closure defined above solves the problem described in section 6.3. If a knowledge base contains the information that birds generally fly but contains no information about colors, we shall be able to conclude that green birds generally fly too.

8 Nonmonotonicity

The notion of rational closure as proposed in section 7.2 models two distinct types of nonmonotonicity. First, for given $\mathbf{K}$, it may well happen that some conditional assertion $\alpha \vdash \beta$ is in $\overline{\mathbf{K}}$ whereas $\alpha \wedge \gamma \vdash \beta$ is not. But a more subtle kind of nonmonotonicity is also at work: if $\mathbf{K} \subset \mathbf{K}'$, it must not be the case that $\overline{\mathbf{K}} \subseteq \overline{\mathbf{K}'}$. One example only of this phenomenon will be described. Consider a knowledge base $\mathbf{K}$ where the propositional variable p does not appear, that contains $q \vdash r$ but that does not preferentially entails $q \vdash \mathbf{false}$. By corollary 3, $\overline{\mathbf{K}}$ contains $q \wedge p \vdash r$. Let now $\mathbf{K}'$ be $\mathbf{K} \cup \{q \wedge p \vdash \neg r\}$. It is easy to check that $q \wedge p \vdash r$ is not in $\overline{\mathbf{K}'}$.

9 Example

Let $\mathbf{K}$ be the famous knowledge base consisting of three assertions:

1. $bird \vdash fly$
2. $penguin \vdash bird$
3. $penguin \vdash \neg\, fly$

The complete description of the rational closure of $\mathbf{K}$ is not an easy task, but let us list some assertions contained in $\overline{\mathbf{K}}$ and some that are not. In the rational closure are : $fly \vdash \neg\, penguin$, $\neg\, fly \vdash \neg\, bird$, $\neg\, fly \vdash \neg\, penguin$, $bird \vdash \neg\, penguin$, $\neg\, bird \vdash \neg\, penguin$, $bird \wedge penguin \vdash \neg\, fly$, $penguin \wedge black \vdash \neg\, fly$, $bird \wedge green \vdash fly$.

The following assertions are not in $\overline{\mathbf{K}}$: $bird \wedge \neg fly \vdash penguin$, $bird \wedge \neg fly \vdash \neg\, penguin$, $penguin \vdash fly$.

10 Open problems and future work

More work has to be done to assess whether the rational closure operation proposed here fully fits our intuition. Better ways of completely describing the rational closure of finite knowledge bases have to be found. The complexity of computing the rational closure should be analyzed and practical heuristics looked for. Families of knowledge bases for which efficient algorithms exist may probably be defined. One could also look for families of knowledge bases with perfect rational extensions but the simplicity and naturalness of the counter-example used in the proof of lemma 2 leaves little hope.

11 Related works

This work stems from a very different motivation than the vast body of work concerned with conditional logic and its semantics that is surveyed in [Nute, 1984]. The main technical difference is that conditional logic considers the nonmonotonic consequence as a new logical connective that is part of the language (here it is part of the meta-language) and that can be embedded inside other connectives and even itself. One of the logical systems, $\mathbf{P}$, studied in this paper turns out to be the flat (i.e. non-nested) fragment of one of the numerous logical systems proposed for conditional logic, introduced by J. Pollock [Pollock, 1976] and studied by J. Burgess in [Burgess, 1981] and by F. Veltman in [Veltman, 1986]. Theorem 1 cannot be derived from the completeness result of [Burgess, 1981] since the latter concerns a extended language and it is not clear that a proof in the extended language may be translated in the restricted one.

This very fragment had been considered by E. Adams in [Adams, 1975] (see also [Adams, 1966] for an earlier version and motivation), where it was given a probabilistic justification. The results of [Lehmann and Magidor,] allow for an easy proof of the completeness result of Adams. Recently J. Pearl and H. Geffner [Pearl and Geffner, 1988] studied a system for nonmonotonic reasoning based on Adams' ideas.

The material on rational reasoning in this paper may be compared with the results of J. Delgrande in [Delgrande, 1987] and [Delgrande, 1988]. The general thrust is very similar but differences are worth noticing. A first difference is in the language used. Delgrande's language differs from this paper's in three respects: his work is specifically tailored to first-order predicate calculus, whereas this work deals with propositional calculus; he allows negation and disjunction of conditional assertions, which are not allowed in this paper; he allows nesting of conditional operators in the language, though his results are good only for unnested formulas. Therefore Delgrande's central completeness result in [Delgrande, 1987], only shows that any proposition in which there is no nesting of conditional operators (let us call those propositions *flat*) that is valid has a proof from the axioms and rules of his system. But this proof may use propositions that are not flat. The completeness results reported here show that valid assertions have proofs that contain only flat assertions.

A second difference is that Delgrande's logical system is different from ours: Delgrande's logic $\mathbf{N}$ does not contain **Cautious Monotony**. Our class of ranked models is more restricted than his class of models: our models are required to obey the smoothness condition and Delgrande's are not. One may also notice that our logic enjoys the finite model property,

but Delgrande's does not. This difference between our two logical systems may sound light when one remarks that many instances of the rule of **Cautious Monotony** may be derived from **Rational Monotony**, and are therefore valid in Delgrande's system **N**. What we mean is that if $\alpha \vdash \beta$ and $\alpha \vdash \gamma$ then, if $\alpha \not\vdash \neg\beta$ one may conclude $\alpha \wedge \beta \vdash \gamma$ by **Rational Monotony** rather than by **Cautious Monotony**. But if $\alpha \vdash \neg\beta$, and therefore $\alpha \vdash$ **false** one cannot conclude. The rule

$$\frac{\alpha \vdash \textbf{false}}{\alpha \wedge \beta \vdash \textbf{false}} \qquad (9)$$

is sound in preferential logic but not in Delgrande's logic.

A third difference is that his definition of the set of conditional assertions entailed by a conditional knowledge base is different from the one presented here, at least at first sight.

12 Acknowledgments

David Makinson suggested to study the rule called here **Rational Monotony** and conjectured that the corresponding family of models was that of ranked models. Discussions with Johan van Benthem helped put this work in perspective and are gratefully acknowledged. Haim Gaifman disproved a hasty conjecture.

References

[Adams, 1966] Ernest W. Adams. Probability and the logic of conditional. In J. Hintikka and P. Suppes, editors, *Aspects of Inductive Logic*. North Holland, Amsterdam, 1966.

[Adams, 1975] Ernest W. Adams. *The Logic of Conditionals*. D. Reidel, Dordrecht, 1975.

[Burgess, 1981] John P. Burgess. Quick completeness proofs for some logics of conditionals. *Notre Dame Journal of Formal Logic*, 22:76–84, 1981.

[Clark, 1978] K. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logics and Data Bases*, pages 293–322. Plenum Press, 1978.

[Delgrande, 1987] James P. Delgrande. A first-order logic for prototypical properties. *Artificial Intelligence*, 33:105–130, 1987.

[Delgrande, 1988] James P. Delgrande. An approach to default reasoning based on a first-order conditional logic: Revised report. *Artificial Intelligence*, 36:63–90, August 1988.

[Gabbay, 1985] Dov M. Gabbay. Theoretical foundations for non-monotonic reasoning in expert systems. In Krzysztof R. Apt, editor, *Proc. of the NATO Advanced Study Institute on Logics and Models of Concurrent Systems*, pages 439–457, La Colle-sur-Loup, France, October 1985. Springer-Verlag.

[Kraus *et al.*,] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. Preferential models and cumulative logics. Accepted for publication in Artificial Intelligence.

[Kraus *et al.*, 1988a] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. Preferential models and cumulative logics. Technical Report TR 88-15, Leibniz Center for Computer Science, Dept. of Computer Science, Hebrew University, Jerusalem, November 1988.

[Kraus *et al.*, 1988b] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. Preferential models and cumulative logics. Submitted for publication, August 1988.

[Lehmann and Kraus, 1988] Daniel Lehmann and Sarit Kraus. Non monotonic logics: models and proofs. In *European workshop on logical methods in artificial intelligence*, pages 58–64, Roscoff (Finistère) France, June 1988.

[Lehmann and Magidor,] Daniel Lehmann and Menachem Magidor. Rational logics and their models: a study in cumulative logics. Submitted.

[Lehmann and Magidor, 1988] Daniel Lehmann and Menachem Magidor. Rational logics and their models: a study in cumulative logic. Technical Report TR 88-16, Leibniz Center for Computer Science, Dept. of Computer Science, Hebrew University, Jerusalem, November 1988.

[Lehmann, 1988] Daniel Lehmann. Preferential models and cumulative logics. In Ehud Shapiro, editor, *Fifth Israeli Symposium on Artificial Intelligence, Vision and Pattern Recognition*, pages 365–381, Tel Aviv, December 1988. Information Processing Association of Israel.

[Makinson, 1989] David Makinson. General theory of cumulative inference. In *Proceedings of the second international workshop on non-monotonic reasoning*. Springer Verlag, January 1989. Lecture Notes in Computer Science, subseries on A.I.

[McCarthy, 1980] John McCarthy. Circumscription, a form of non monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[McDermott and Doyle, 1980] Drew McDermott and John Doyle. Non-monotonic logic I. *Artificial Intelligence*, 25:41–72, 1980.

[Nute, 1984] Donald Nute. Conditional logic. In Dov M. Gabbay and Franz Guenthner, editors, *Handbook of Philosophical Logic*, chapter Chapter II.8, pages 387–439. D. Reidel, Dordrecht, 1984.

[Pearl and Geffner, 1988] Judea Pearl and Hector Geffner. Probabilistic semantics for a subset of default reasoning. TR CSD-8700XX, R-93-III, Computer Science Dept., UCLA, March 1988.

[Pollock, 1976] J. Pollock. *Subjunctive Reasoning.* D. Reidel, Dordrecht, 1976.

[Reiter, 1980] Raymond Reiter. A logic of default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Reiter, 1987] Raymond Reiter. *Nonmonotonic Reasoning*, volume 2 of *Annual Reviews in Computer Science*, pages 147–186. Annual Reviews Inc., 1987.

[Shoham, 1987] Yoav Shoham. A semantical approach to nonmonotonic logics. In *Proc. Logics in Computer Science*, pages 275–279, Ithaca, N.Y., 1987.

[Shoham, 1988] Yoav Shoham. *Reasoning about Change.* The MIT Press, 1988.

[Tarski, 1956] Alfred Tarski. *Logic, semantics, metamathematics. Papers from 1923–1938.* Clarendon Press, Oxford, 1956.

[Vardi, 1988] Moshe Y. Vardi, editor. *Proceedings of the Second Conference on Theoretical Aspects of Reasoning About Knowledge*, Pacific Grove, California, March 1988. Morgan Kaufmann.

[Veltman, 1986] Frank Veltman. *Logics for Conditionals.* PhD thesis, Filosofisch Instituut, Universiteit van Amsterdam, 1986.

Analogy as a Constrained Partial Correspondence Over Conceptual Graphs

Debbie Leishman

Knowledge Science Institute & Department of Computer Science
University of Calgary
Alberta, Canada T2N 1N4

Abstract

This paper describes an analogical tool based on the formation of partial correspondences between knowledge structures represented as conceptual graphs. The formation of correspondences is controlled by evaluation criteria for ordering analogies by plausibility that relate naturally to operations on conceptual graphs. The tool and its underlying theory are tested empirically against critical examples of analogical reasoning in the literature. To clarify relations between the examples, reasoning patterns and knowledge representation scheme, an object-oriented implementation is used whose operation is simple and transparent.

1 Introduction

Analogical reasoning is based on the formation of a plausible partial correspondence between knowledge structures. The problem for computational emulations of analogy is to determine what these knowledge structures are and what constitutes a plausible correspondence. There are a variety of applications of analogical reasoning which can help direct the search for a solution to this problem. Analogy is used in metaphor understanding [Indurkhya, 1985, Lakoff and Johnson, 1980], in scientific exploration [Gentner, 1987, Hesse, 1966], in argumentation and rhetoric [Hesse, 1966], in schema abstraction for problem solving [Gick and Holyoak, 1983] and for teaching [Murphy et al., 1963]. Examples of these different applications of analogy, particularly problematic cases, can serve as test cases for a computer emulation of analogy.

This paper presents an analogical tool which embodies critical functions of analogical reasoning derived from the analysis of previous research in the field. It is shown that the choice of a conceptual graph model for knowledge representation ensures that these functional requirements are met and also brings with it certain evaluation criteria which constrain the search for analogies. The tool is described in three ways, firstly in terms of the conceptual graph formalism, secondly in terms of the evaluation criteria contained within it, and thirdly, in terms of the basic algorithm. The final section discusses how well the analogical tool performs on examples from the literature and shows it to be effective in the derivation of analogies and in reasoning based on them. The simple modular structure of the algorithm clarifies the relations between the argument forms of analogical reasoning and general properties of the knowledge representation schemes used.

2 Analysis of Prior Studies of Analogy

Analysis of colloquial connotations and psychological processes of analogy has generated research which can be classified as follows:

1) Abstract theories of analogy;
2) Psychological models of analogy;
3) Evaluation criteria for analogy.

Analyzing works in each of these areas results in a definition of critical functions essential to the notion of analogy. These critical functions can be found in many computational emulations of analogy although often they are not explicitly stated.

Abstract theories view analogy as an algebraic or logical process. Comparison of works in this area reveals the common theme that analogy is the

formation of a plausible partial correspondence between parts of the participating analogues. Indurkhya [1985], for example, describes a theory of analogy and metaphor called Approximate Semantic Transference which forms AT-MAPS. These maps represent an approximately coherent partial mapping of terms between source and target domains. Gaines and Shaw [1982] explicate analogy as a partial correspondence between systems represented as mathematical categories. Russell's use of determinations [Russell, 1987] also requires the formation of a plausible partial correspondence between terms.

Psychological models of analogy attempt to account for how and in what circumstances people reason from analogies. Comparison of works in this area reveals the common theme that a critical part of any model of analogy is the knowledge that is used and how it is represented. Gentner's Structural Mapping Theory [1983] is based on a principle of systematicity which attempts to maintain chains of relations when forming correspondences between domains. The choice of higher order predicate calculus as the knowledge representation scheme aids in detection of these chains. Lakoff and Johnson [1980] argue that the human conceptual system is comprised of a limited number of non-metaphorical base concepts supplemented by metaphorical concepts based on experiential gestalts. These gestalts can be thought of as analogues, and a partial correspondence formed over terms they contain allows us to understand metaphors such as ARGUMENT IS WAR. From this psychological model, Lakoff and Johnson note that the most difficult part of constructing a computational emulation of metaphor understanding will be in properly representing the knowledge contained in the experiential gestalts.

Evaluation criteria for analogy focus on the problem of what is a plausible analogy—why one analogy is seen as more plausible than another. Comparison of works in this area reveals the need to control the formation of plausible partial correspondences through evaluation criteria which relate to psychological notions of plausibility and the applicability of analogies. For example, evaluation criteria which use generalization/specialization hierarchies are central to the CYC project being conducted at MCC [Lenat et al, 1986], as well as to the work of Hofstadter and Mitchell on analogical reasoning [1988]. Criteria based on connectivity can be found in Indurkyha's [1985] AST theory and in Gentner's [1983] systematicity principle. Winston's [1981] early paper on analogical reasoning contains many useful criteria such as the use of case relations and matching of terms which share the same relations between them; this latter criterion is also evident in Gentner's work.

This three-way analysis leads to the following critical functions which are necessary for a computational emulation of analogy:

1) Algorithms for finding partial correspondences between abstract structures;
2) Semantic schemas for representing knowledge of the domains involved in the analogies;
3) An evaluation relation for partially ordering analogies by plausibility.

The analogical tool described in this paper implements each of these functions as well defined modules. The evaluation criteria arise naturally from the knowledge representation scheme chosen, namely conceptual graphs. These criteria include those used by other researchers. The resulting algorithm clearly defines the rationale for the analogies formed, and is readily tested on examples of analogy taken from the literature.

3 Conceptual Graphs

The semantic schema requirement above is satisfied in this work by the use of the conceptual graph model as described by Sowa [1984]. A single conceptual graph states a proposition and contains concept nodes, relation nodes, and arcs between them. This structure allows analogues to be represented as finite bipartite graphs. Concept nodes represent entities, attributes, states and events while relation nodes specify relationships between concepts and include case relations among others [Sowa, 1984]. Each concept in a conceptual graph is part of a lattice of concepts. This lattice supports inheritance, and concepts higher in the lattice are more general than lower ones. Concepts to be used in a domain must be specified beforehand.

Graphs can also express more complex sentences that contain individuals, modality, nested propositions and co-references as shown in Figure 1. The sentence for this graph is shown at the top. Co-reference links are shown with dashed lines and the nested propositions are shown by the boxes surrounding subgraphs. Concepts in a conceptual graph can be generic, such as [Person] or can refer to

"Sam thinks that the house has a kitchen and that Ivan believes that there is a cat in the kitchen"

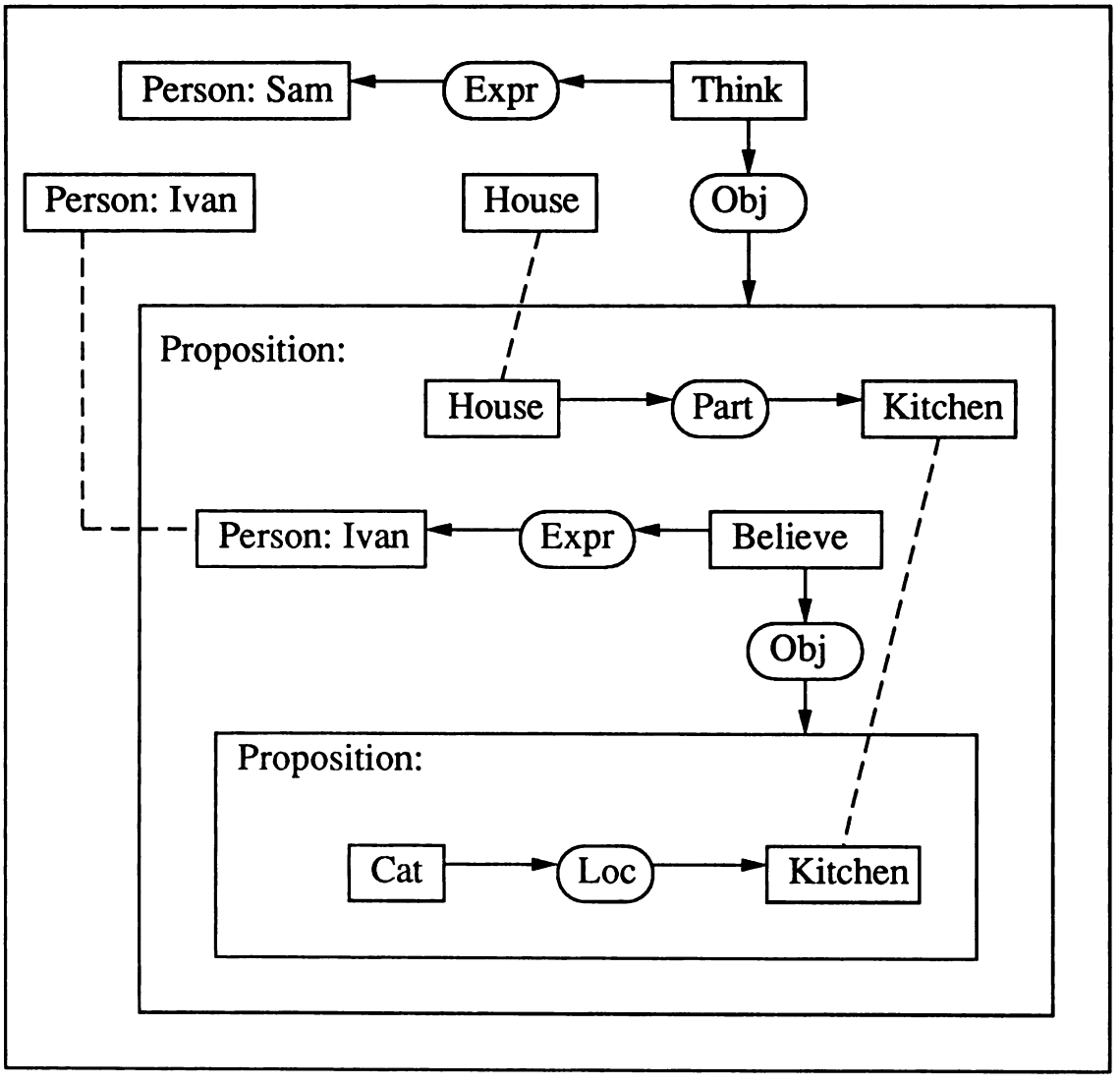


Figure 1 Conceptual graph for a complex sentence

particular individuals such as [Person: Sam]. This concept refers to a specific individual *Sam* of type *Person* where *Sam* is specified in the referent field of the concept. The modal concepts *Think* and *Believe* in Figure 1 take nested conceptual graphs as their objects. These nested graphs are *Propositions* whose referent field is the set of conceptual graphs being asserted. They are also referred to as context boxes and a box with no type label defaults to a proposition (context) box. Also present in this figure are co-reference links which show anaphoric references and express an equality relation between concepts. Co-reference links often join pronouns to their referents or refer to a concept previously mentioned. In Figure 1 for example the phrase, "Sam thinks that the house has a kitchen," refers to a house that Sam already knows about. This requires a co-referent to house outside the context of the house having a kitchen.

Each conceptual graph is also embedded in a larger semantic net which contains background knowledge that enriches the meaning of the graph.

3.1 Specialization and Generalization

The conceptual graph model defines specialization operations used for selectional constraints on sentence formation and generalization operations for logic and model theory. Specialization of a conceptual graph can be done in two ways. First a particular concept can be restricted, either by addition of an individual to a generic concept or by replacing a concept with one of its subtypes from the concept lattice. Second, a conceptual graph can be specialized by making the graph more extensive. This amounts to adding more concepts and relations to a graph. Rules for when and how these specialization operations take place are specified by the conceptual graph model [Sowa, 1984]. These specialization rules and operations are not truth preserving but enforce constraints on sentence formation.

Generalization of conceptual graphs is truth preserving and if a graph u is a specialization of a graph v, written u <= v, then v is a generalization of u. Generalization of conceptual graphs defines a partial ordering called a generalization hierarchy. Graphs can be generalized in three ways:

1) Any subgraph is a generalization of the original graph;
2) Replacing a type label with a supertype generalizes a graph;
3) Deleting an individual from the referent field generalizes a graph.

These specialization and generalization operations have important uses in analogy formation and analogical reasoning.

3.2 Conceptual Graphs for Analogy

There are three reasons why conceptual graphs are a good choice for the knowledge representation scheme of an analogy emulation program:

1) They have a direct mapping to predicate calculus and thus can be incorporated into any reasoning system based on logic [Sowa, 1984];
2) They contain constraints which constitute evaluations for forming partial correspondences as well as an evaluation relation for partially ordering analogies by plausibility;
3) Their straightforward mapping to and from natural language allows for a linguistic description of analogues.

4 Implementing the Analogical Tool

This section outlines the implementation of the analogical tool in terms of the conceptual graph operations used and the resulting specifications for algorithms.

4.1 Minimal Common Generalizations

In the conceptual graph formalism, the natural partial correspondence on which to base the formation of an analogy is the *minimal common generalization* of the graphs corresponding to two or more analogues. A common generalization specifies a graph which is a generalization of all analogues in an analogy. There can be more than one of these generalizations and as stated previously, they will be ordered in a generalization hierarchy. Figure 2 gives a very simple example of three ordered common generalizations which can be formed from the two analogues shown.

The lowest generalization on the page is the minimal common generalization. It is the most specific of the three, first in concept restriction (from *Animate* to *Person*) and second in extent (it contains the maximal number of nodes and arcs). A minimal common generalization is a generalization as low as possible in the generalization hierarchy —such that no other is more specific.

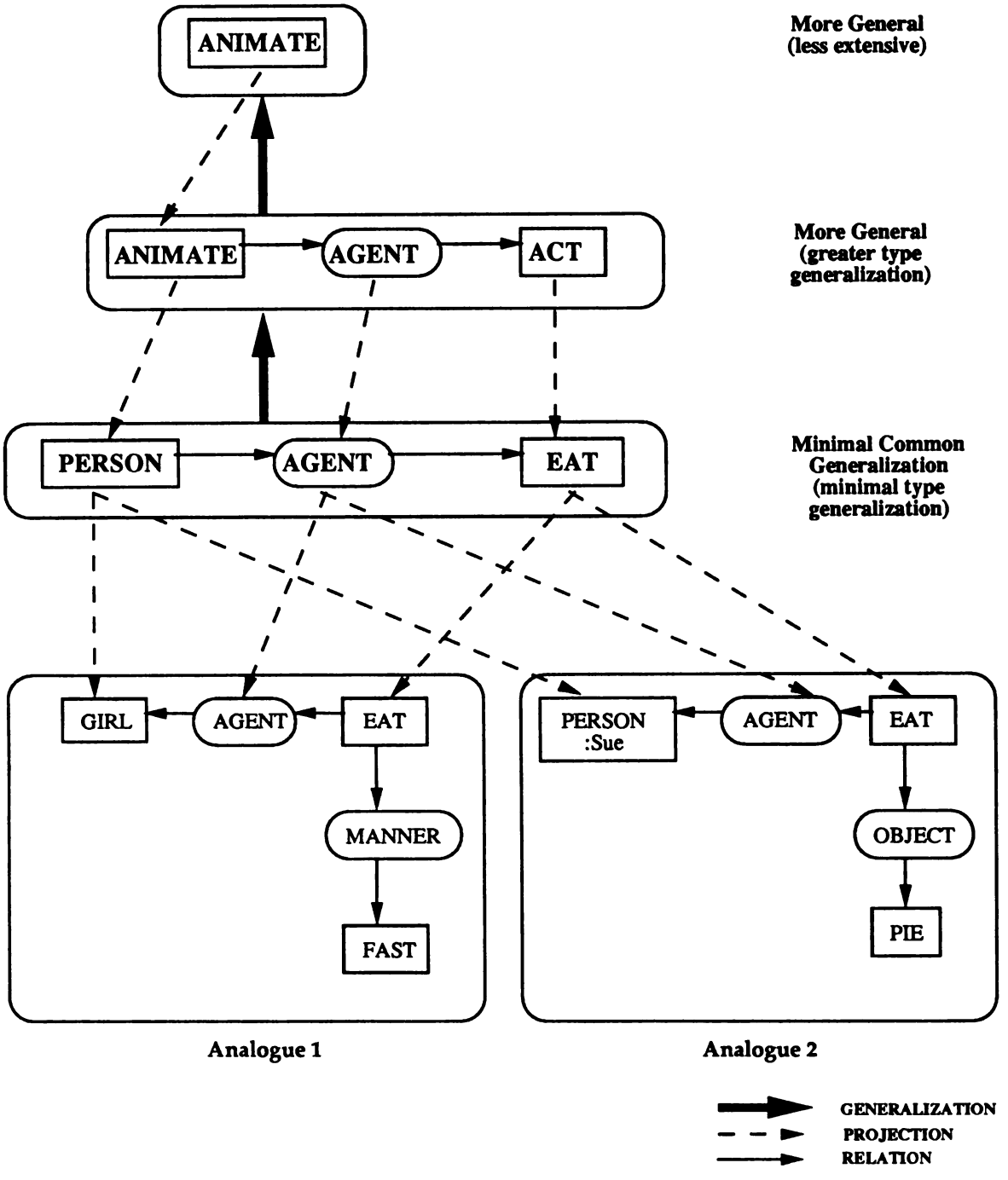


Figure 2 Forming Common Generalizations

It is these minimal common generalizations that the analogical tool derives. They satisfy requirements 1) and 3) specified in Section 2, that is of providing: a basis for algorithms for finding partial correspondences; and an evaluation relation for partially ordering analogies by plausibility.

A minimal common generalization:

- Is maximally extensive since the generalization chosen for each of the corresponding concepts is the lower bound of all possible common generalizations;
- Preserves linkages in the graphs which can be seen as minimal *structural* criterion for plausibility;
- Is truth preserving, a necessary *logical* criterion for plausibility;
- Minimizes conflicts in the type hierarchy, a minimal *semantic* criterion for plausibility.

There will typically be many generalizations of two analogues, a subset of which will be minimal. The analogical tool uses four formation evaluations to search for generalizations, and two general ordering evaluations to establish which of these are minimal. These are described in the following sub-sections.

4.2 Formation Evaluations

Four formation evaluations arise from the generalization criteria, and reduce search in the analogy operation. These evaluations are used to form the common generalizations.

The first two were presented above; firstly the requirement of the same relations between two pairs of potentially corresponding concepts; and secondly the requirement of type comparability. Two concepts will be comparable if there exists a common generalization for them which is not the absurd type. For example, an event and an attribute will not be comparable concepts, but two entities will be. The third and fourth formation evaluations emerge as analogues become compound graphs with co-reference links and nested context boxes.

Co-reference links form chains of equality between concepts in a graph. For the third evaluation, these chains of equality must be maintained when forming the common generalizations and putting concepts into correspondence.

The fourth evaluation requires that referents of corresponding proposition concepts (i.e. graphs) must be compatible. This means that if projections put two proposition boxes into correspondence in the analogy, then their referent subgraphs are candidates to be put into correspondence only with each other.

4.3 Ordering Evaluations

The formation evaluations above are a consequence of specifying an analogy as a common generalization. Specification of the more plausible analogies as *minimal* common generalizations has implications for ordering them. A minimal common generalization is maximally extensive. Because of the partial nature of the generalizations formed in analogy, this maximal extent becomes a measure of connectivity. The first general ordering evaluation is a measure of this connectivity.

This first evaluation, connectivity, can be subdivided into three specific components. The first, **Extent**, measures how much of the target domain has been matched with the source domain. A measure of 0 reflects a total match. This ensures that only the most extensive analogies are considered. The second component evaluates the **Kind-Of** connectivity that results in the most extensive analogies. Preference is given to long chains of connectivity and a high score here will indicate more connectivity and less fragmentation. The third component is a measure of **Extendability**. Given two equal analogies as ordered by the first two components of connectivity, the one with a greater capability of being extended through analogical reasoning is preferred. An example of this measure appears in the section on testing the analogical tool.

Minimal common generalizations are also those which choose the lowest common supertypes for each pair of corresponding concepts as defined by the concept lattice. A higher preference is given to analogies with the lowest semantic distance between corresponding concepts. The second general ordering evaluation says that for two analogies which are equally connected, the better analogy is determined by minimizing the distance climbed on the concept lattice for each pair of comparable concepts. This amounts to measuring the semantic distance of concepts according to this lattice. Concepts which are further apart correspond to weaker analogies.

4.4 Correspondence and Projections

The formation of a minimal common generalization of two analogues involves putting the concepts and relations in one analogue in correspondence with those in the other, and hence in the generation of an analogy. This correspondence again has a natural interpretation in conceptual graphs as arising from a pair of *projections*.

As shown in Figure 2, formation of a common generalization results in projections from the generalization into each of the analogues (these are shown by the dashed lines). A projection maps graphs at higher levels of the generalization hierarchy into ones at lower levels. A projection is defined as follows [Fargues, 1986]:

Let **u** and **v** be two conceptual graphs. If **u <= v**, there exists a subgraph **u'** joined with some additional edges. The induced graph **u'** is called the projection of **v** in **u**. A graph **v** may have several distinct projections in the same given graph **u**.

The projection operation consists of finding a subgraph **u'** of **u** which satisfies the following conditions:

1) The conceptual relations in u' and v are identical;
2) The concepts c1, . . ., cn of u' are some restrictions (i.e. specializations) of the corresponding concepts d1, . . ., dn of v, as defined by the concept lattice;
3) If a relation r links two concepts di and dj in v, then it also links the concepts ci and cj in u'.

4.5 Object-Oriented Formulation of the Algorithm

The analogical tool is written in the object oriented language Smalltalk [Goldberg and Robson, 1983] in such a way that each of the formation and ordering evaluations exist as well defined, identifiable modules. The basic algorithm searches for analogies, with the formation evaluations serving as constraints that prune branches of the search tree; and the ordering evaluations ordering the analogies. Input consists of two analogues expressed as conceptual graphs. Output consists of displays of the analogies as lists of concept correspondences, together with their measures.

As analogies are being formed, potential concept matches are considered. These must pass all formation evaluations before the concepts can be put into correspondence with each other. For example, the requirement of same relations between concepts is a very strong constraint and permits only agents to match to agents, objects to objects, etc.

Only the most extensive analogies as measured by the **Extent** connectivity metric are considered further. These are then ordered using the **Kind-Of** connectivity metric. Any ties in ordering at this point are broken using the semantic distance of concepts metric.

At this point, if an analogy is **Extendable**, analogical reasoning can take place. This reasoning attempts to maximize the connectivity measure and minimize the semantic distance measure while adhering to the formation evaluation constraints. Examples of the tool in operation in the following section clarify these points.

5 Testing the Analogical Tool

The analogical tool has been tested on various examples of analogy from the literature. Examples have been taken from the CYC project [Lenat et al, 1986], from Gentner's work [1983], from Indurkyha's PhD thesis [1985], from Russell's PhD thesis [1985] and from Gick and Holyoak [1983]. Two of these examples are presented here. They show that the analogical tool performs effectively, using evaluations which are a result of the knowledge representation scheme chosen.

5.1 Example I

The first example, shown in Figure 3, is taken from work on schema abstraction by Gick and Holyoak [1984] and will be used in two ways by the analogical tool: first to form a general problem solving principle, and second to perform simple analogical reasoning.

In the first case, two analogous problems and their solutions are presented. An analogy is drawn between the two domains then abstracted to obtain a general problem solving principle. This abstraction can later be used to solve similar problems. Gick and Holyoak use as an example a military problem and an analogous radiation problem.

In the military problem, a commander wants to capture a fortress using a large army but cannot send the entire army down one road because they will be noticed. The solution is to send small parts of the army along many roads simultaneously. These sentences are represented in the four conceptual graphs

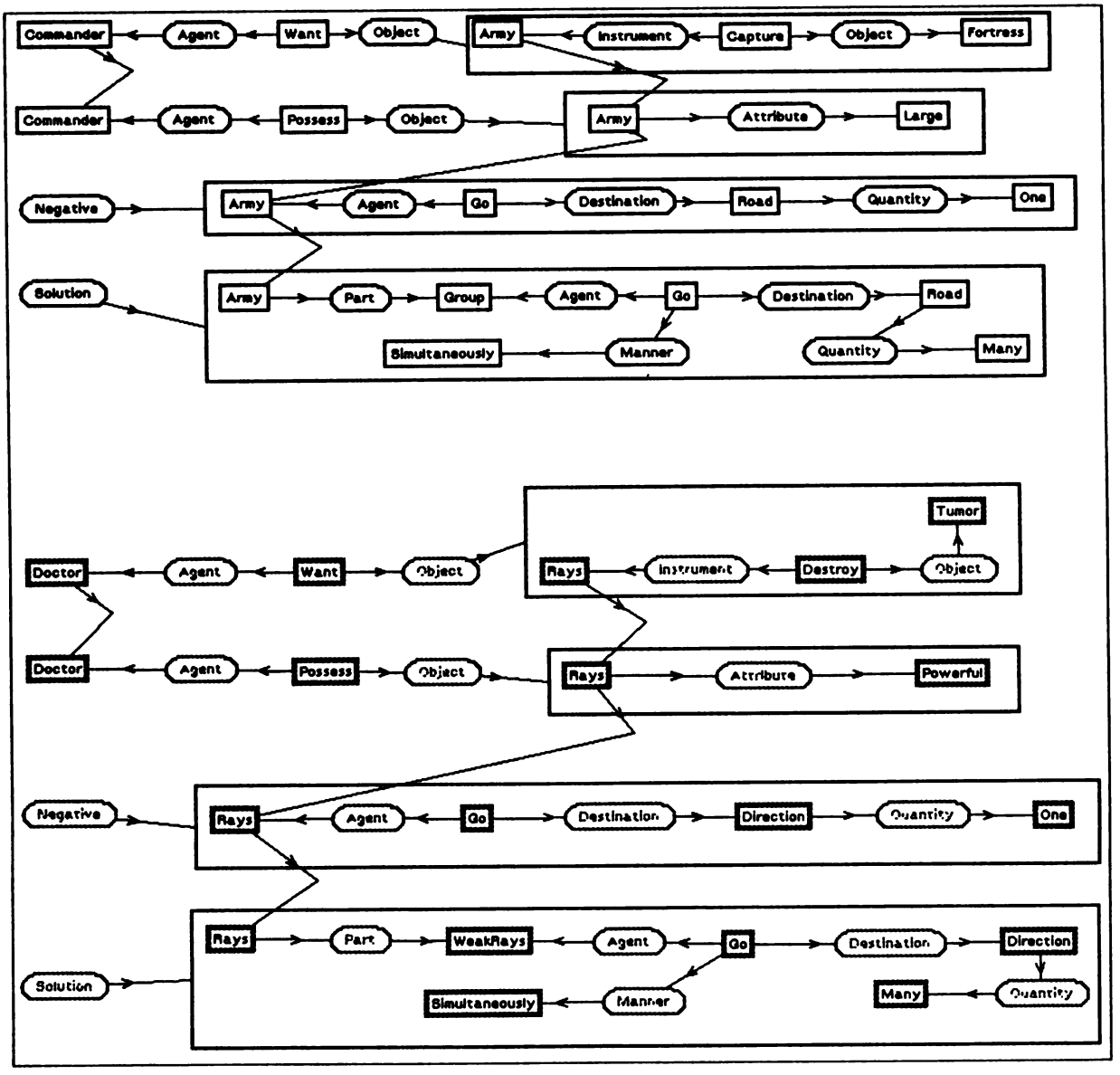


Figure 3 Example of Schema Abstraction

at the top of Figure 3. Co-referent links are shown by the very angled arcs connecting concepts with the same name. Context boxes representing propositions are the boxes which contain subgraphs within them.

In the analogous radiation problem, a doctor wants to use rays to destroy a tumor. The rays cannot all come from one direction without killing the patient. The analogous solution is to send low intensity rays from multiple directions simultaneously. These sentences are represented in the bottom four conceptual graphs containing dashed nodes in Figure 3.

These two situations as represented are given to the analogical tool, which forms the same analogy as that presented by Gick and Holyoak. The following concept correspondences are formed by the tool:

Doctor.....Commander	Rays.....Army
Destroy.....Capture	Tumor.....Fortress
Want.....Want	Possess.....Possess
Powerful.....Large	Go.....Go
Direction.....Road	One.....One
WeakRays....Group	Many.....Many

The tool also automatically forms an abstraction similar to that described by Gick and Holyoak, by using the concept lattice to find common generalizations of the concepts. The general problem solving principle formed by the tool is:

A Person wants to use a Mobile Entity to perform a Violent act on a Stationary Entity.

A Person has a Mobile Entity with an Attribute of Strength.

The Mobile Entity cannot go along one Path.

The Solution is for Parts of the Mobile Entity to Go along Many Paths Simultaneously

Note that, although the sentences of the analogues appear in the same order in Figure 3, the tool will still find the analogy if the order of the sentences is changed.

This example can also be used to show how the tool can perform simple analogical reasoning. In this case the analogues are the same with the exception that the solution to the radiation problem is not known. This is represented as an empty *Proposition* node attached to the *Solution* relation. When the tool reports the correspondences resulting from the best analogy formed, the two *Solution* propositions are matched together. With substitution of corresponding terms inside the solution to the military problem, a solution to the radiation problem is realized: to have parts of the rays go in many directions simultaneously.

5.2 Example II

Example two, shown in Figure 4, is taken from Gentner's description of a heat-water analogy used by the French thermodynamicist Carnot [Gentner, 1987]. This example is very significant because it contains a pitfall for over-simplistic analogy formation processes. The problem arises because two equal matches occur. An analogy formation algorithm must be able to determine which is the more relevant one. Use of the *systematicity principle* in Gentner's Structure-Mapping program results in the correct choice. For the analogical tool described in this paper, the more relevant match is correctly detected by the **Extendability** measure.

At the top of Figure 4 is a representation a person might have of a water flow situation. The person knows that if the pressure being exerted on water in a beaker is greater than the pressure being exerted on water in a vial, and if there is a fluid path from the beaker to the vial via a pipe, then this will result in a flow of the water through the pipe from the beaker to the vial. The person will also have other knowledge about the situation such as the diameter of the beaker being greater than the diameter of the vial.

The graph at the bottom of Figure 4, represented by dashed nodes with connecting arcs, is a description of what that same person might know about heat flow. In this case the person knows that the coffee is hotter than the ice-cube and that heat will flow from the coffee to the ice-cube. The person does not know that the heat flow is a result of the difference in heat between the ice-cube and the coffee.

These two analogues are presented to the analogical tool. In this example the tool forms two analogies with equal measures. Both analogies have formed the same correspondences between concepts except that in one *Temperature* has been matched to *Pressure*, and in the other *Temperature* has been matched to *Diameter*. Both of these analogies have the same connectivity rating as given by the **Extent** and **Kind-Of** connectivity measures.

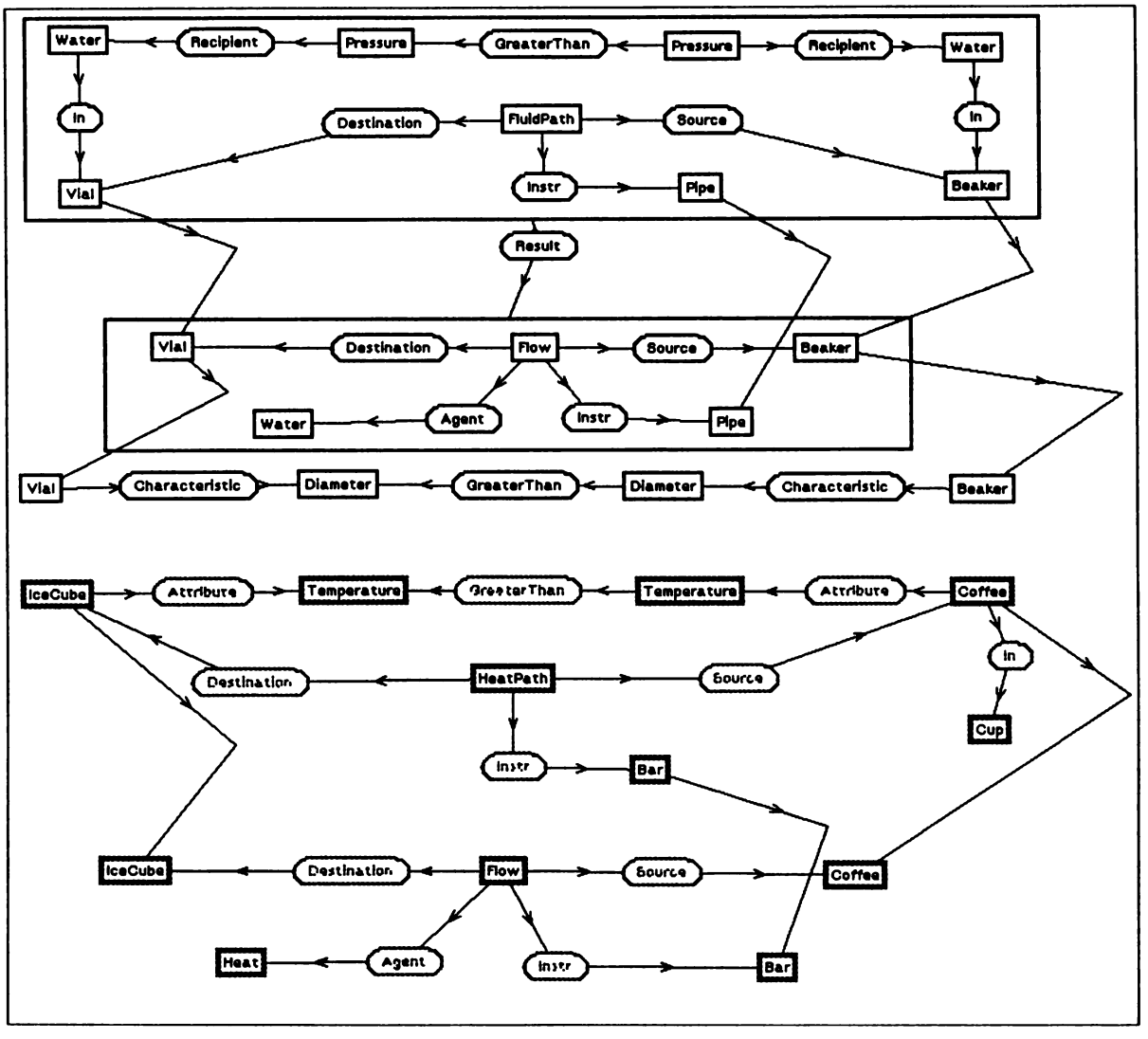


Figure 4 Example of A Heat-Water Analogy

Using the **Extendability** measure in the analogy which matches *Temperature* with *Pressure*, it is possible to hypothesize both of the context boxes from the source water domain as well as the result relation between them over to the target heat domain. This hypothesis will be consistent with all formation evaluation criteria. In contrast, the analogy which has matched *Temperature* with *Diameter* can only hypothesize the context box which surrounds the flow concept and its relations. It is not possible to hypothesize the other context box as existing without breaking the proposition evaluation criteria. Thus the **Extendable** ordering evaluation not only answers whether an analogy is extendable or not but can also help with decisions about which hypotheses are best to make when performing analogical reasoning.

The result of this use of **Extendability** is very similar to that obtained by Gentner's use of the systematicity principle. The difference is that rather than being an explicit driving principle like systematicity, the **Extendability** measure is more of an emergent property. This property is part of a system which has a strategy of maximizing ordering evaluations while adhering to the formation evaluation constraints.

6 Summary and Conclusions

This paper has described an analogical tool based on the formation of partial correspondences between knowledge structures represented as conceptual graphs. Evaluation criteria have been derived for ordering analogies by plausibility that relate naturally to operations on conceptual graphs. These have been tested empirically against critical examples of analogical reasoning in the literature.

The algorithms are written in Smalltalk to give a tool whose operation is simple and transparent so the essential relations between the examples, reasoning patterns and knowledge representation scheme are apparent.

The use of conceptual graphs for knowledge representation has been beneficial:

i) in allowing examples of analogical reasoning in the literature expressed in natural language to be simply translated to a computational representation;

ii) in providing a principled algebraic formulation of the essential operations of analogy formation and analogical reasoning that relates simply and naturally to the semantic constraints found in cognitive studies of analogy.

The notions of *analogy* and the relative *plausibility* of analogies are cognitively rich and it is not clear that any computational framework can capture them in full. There is certainly a very strong dependence on the underlying knowledge representation scheme that is used. The results of this study indicate that conceptual graphs naturally support the mathematical operations of forming partial correspondences and provide evaluation criteria that are strong enough to result in plausible analogies that correspond to psychological expectations.

Acknowledgements

This work has been partially supported by the Natural Sciences and Engineering Research Council of Canada. I am grateful to the anonymous referees for their perceptive criticism; to John Sowa for helpful comments and access to his own research; and to Brian Gaines, Rosanna Heise, Brent Krawchuk, Mildred Shaw and Ian Witten for critical suggestions that have improved this paper.

References

[Fargues, 1986] Jean Fargues. Conceptual Graphs for Semantics and Knowledge Processing. IBM Journal of Research and Development. Vol. 30 No. 1 1986.

[Gaines and Shaw, 1982] Brian Gaines and Mildred Shaw. Analysing Analogy. Trappl, R. Ricciardi, L. and Pask, G. Editors. Progress in Cybernetics and Systems Research Vol. IX. pp. 379-386. Washington: Hemisphere. 1982.

[Gentner, 1983] Dedre Gentner. Structure-Mapping: a Theoretical Framework for Analogy. Cognitive Science 7. 1983. pp. 155-170.

[Gentner, 1987] Dedre Gentner. Mechanisms of Analogical Learning. Report No. UIUCDCS-R-87-1381. Department of Computer Science. University of Illinois at Urbana-Champaign. 1987.

[Gick and Holyoak, 1983] Mary Gick and Keith Holyoak. Schema Induction and Analogical Transfer. Cognitive Psychology 15. 1983. pp.1-38.

[Goldberg and Robson, 1983] Adele Goldberg and David Robson. Smalltalk-80 The Language and its Implementation. Addison Wesley Publishing Co. Reading, Ma. 1983.

[Hesse, 1966] Mary Hesse. Models and analogies in Science. Notre Dame, IN: University of Notre Dame Press. 1966.

[Hofstadter and Mitchell, 1988] Douglas R. Hofstadter and Melanie Mitchell. Concepts, Analogies and Creativity. Proceedings of CSCSI'88: Seventh Biennial Conference of the Canadian Society for Computational Studies of Intelligence. pp.94-101. 1988.

[Indurkhya, 1985] Bipin Indurkhya. A Computational Theory of Metaphor Comprehension and Analogical Reasoning. PhD thesis, University of Massachusetts, 1985.

[Lakoff and Johnson, 1980] George Lakoff and Mark Johnson. The Metaphorical Structure of the Human Conceptual System. Cognitive Science 4. 1980. pp. 195-208.

[Lenat et al., 1986] Doug Lenat, Maryank Prakash and Mary Shepherd. CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks. The AI Magazine. pp. 65-85.

[Murphy et al, 1963] Glenn Murphy, David J. Shippy and H. L. Luo. Engineering Analogies. Iowa State University Press. Ames, Iowa. 1963.

[Russell, 1987] Stuart Jonathan Russell. Analogical and Inductive Reasoning. PhD thesis. Stanford University. 1987.

[Sowa, 1984] John Sowa. Conceptual Structures. Addison Wesley Publishing Co. Reading Ma. 1984.

[Winston, 1981] Patrick Winston. Learning and Reasoning by Analogy. Communications of the ACM Vol. 23 No. 12 1980.

Between Circumscription and Autoepistemic Logic*

Vladimir Lifschitz
Stanford University
Stanford, California

¡*Modality*—si, *modal logic*—no!
—John McCarthy, personal communication

Abstract

We introduce a modification of circumscription which is in many ways similar to autoepistemic logic, although it does not use modal operators. Traditional "minimizing" circumscription can be viewed as a special case of the new "introspective" version. At the same time, introspective circumscription allows us to represent the forms of nonmonotonic reasoning that correspond to the use of nonnormal defaults.

1 Introduction

Among the available mathematical models of nonmonotonic reasoning, the definition of circumscription [McCarthy, 1980], [McCarthy, 1986] stands out as particularly economical in its use of logical tools. Unlike its main competitors, circumscription is not even a new logic; it is merely a syntactic transformation of formulas that allows us to formalize nonmonotonic reasoning in classical logic.[1]

On the other hand, default logic [Reiter, 1980] and autoepistemic logic [Moore, 1985] provide some expressive possibilities that seem to have no counterparts in circumscription. Circumscribing a predicate P corresponds to the use of the default

$$\frac{:\neg P(x)}{\neg P(x)}$$

("If it is consistent that $\neg P(x)$, then $\neg P(x)$"). This default has a very special form: It is a "normal default without a prerequisite." The corresponding autoepistemic axiom is

$$P(x) \supset \mathrm{L}P(x) \qquad (1)$$

("$P(x)$ only if I believe that $P(x)$"). Such "normal" axioms are, syntactically, very special also.

Recent work on applications of formal nonmonotonic reasoning shows that nonnormal defaults and nonnormal autoepistemic axioms may be quite useful. In [Gelfond, 1987], general logic programs are translated into autoepistemic theories by inserting the modal operator L after each negation. This translation provides a simple and elegant declarative semantics for a wide class of logic programs. In [Morris, 1987], [Gelfond, 1989] and [Morris, 1989], nonnormal defaults and corresponding autoepistemic axioms are used for eliminating unwanted models in nonmonotonic theories of action ("the Yale shooting problem"). These important ideas cannot be reformulated in terms of circumscription in any obvious way.

This discussion leads us to the problem of developing new approaches to formalizing nonmonotonic reasoning, that will combine the attractive features of circumscription, on the one hand, and of default logic and autoepistemic logic, on the other. In the best possible world, a counterpart of nonnormal defaults would be available, as well as the full power of circumscription, while modelling nonmonotonicity would be based on a simple syntactic transformation of formulas of classical logic.

An important step in this direction is made in [Perlis, 1988], where the definition of "autocircumscription" is proposed. It is formally similar to the definition of circumscription. But the new concept "is not really circumscription at all, in the sense that it does not aim at minimizing extensions." Instead, it is "related to self knowledge, and especially to negative introspection."

In this paper we introduce another modification of circumscription along the lines of Perlis's work. We call this form of circumscription "introspective." The main difference between our definition and Perlis's autocircumscription is that our formalism includes both negative and positive introspection. As a result, it has some interesting new properties. First, many versions of McCarthy's "minimizing" circumscription turn out to be equivalent to special cases of introspective circumscription. Second, introspective circumscription is

*This research was partially supported by DARPA under Contract N00039-84-C-0211.

[1] [McCarthy, 1980], Remark 1. "Classical" means here *classical second-order*. Although ill-reputed in the automated deduction community, second-order formulas have always been favorite instruments in foundational studies. Such important postulates as the axiom of induction and the completeness axioms in real arithmetic and geometry are expressed by second-order formulas.

in many ways similar to autoepistemic logic, and has similar applications to logic programming and commonsense reasoning, although it does not use modal operators.

The new form of circumscription is defined in Section 2, and its model-theoretic meaning is discussed in Section 3. Then we study the relation of this theory to the usual "minimizing" circumscription (Section 4) and apply the introspective approach to the semantics of logic programs (Section 5) and to formalizing commonsense reasoning (Section 6).

2 Introspective Circumscription

2.1 Definition

Consider a second-order language with finitely many predicate constants $P_1, \ldots, P_n$, of arities $k_1, \ldots, k_n$. Extend it by adding, for each $i = 1, \ldots, n$, a new predicate constant LP_i, of arity k_i. We will write P for $P_1, \ldots, P_n$, and LP for $LP_1, \ldots, LP_n$.

The formula $LP_i(...)$ reads: $P_i(...)$ *is believed*. Our choice of notation LP_i is related to the modal operator L in autoepistemic logic. But the language we use is not a modal language; it is simply a second-order language in which predicate constants come in pairs, P_i and LP_i.

For any sentence $A(P, LP)$, its *introspective circumscription* $A^I(P, LP)$ is the formula

$$A(P, LP) \wedge \bigwedge_{i=1}^{n} \forall x^i [LP_i(x^i) \equiv \forall p(A(p, LP) \supset p_i(x^i))]. \quad (2)$$

Here p is a tuple of n distinct predicate variables $p_1, \ldots, p_n$, such that the arity of p_i is k_i, and x^i is a tuple of k_i distinct object variables.

Informally, we think of A as the conjunction of all beliefs that an agent has (i) about the predicates P_i and (ii) about his beliefs about the predicates P_i. The conjunctive term

$$\forall x^i [LP_i(x^i) \equiv \forall p(A(p, LP) \supset p_i(x^i))]$$

in (2) expresses the "rationality" of the agent: he believes that $P_i(x^i)$ holds if and only if this follows from his beliefs A. This equivalence will be called the *introspection condition* for P_i. The implications left-to-right and right-to-left represent "negative" and "positive" introspection.

2.2 Special Cases and Examples

In the following examples $n = 1$, so that P is a single predicate, and $A^I(P, LP)$ has the form

$$A(P, LP) \wedge \forall x [LP(x) \equiv \forall p(A(p, LP) \supset p(x))].$$

First consider the case when A does not contain LP. Then A^I can be written as

$$A(P) \wedge \forall x [LP(x) \equiv \forall p(A(p) \supset p(x))].$$

The introspection condition is in this case an explicit definition of LP.

Example 1. If A is $P(a)$, then A^I is equivalent to

$$P(a) \wedge \forall x(LP(x) \equiv x = a)$$

(a is the only object about which I believe that it satisfies P).

Example 2. If A is $P(a) \vee P(b)$, then A^I is

$$[P(a) \vee P(b)] \wedge [a = b \supset \forall x(LP(x) \equiv x = a)] \wedge [a \neq b \supset \forall x \neg LP(x)].$$

On the other hand, if A does not contain P, then A^I is

$$A(LP) \wedge \forall x [LP(x) \equiv \forall p(A(LP) \supset p(x))].$$

In the presence of the term $A(LP)$, the right-hand side of the introspection condition can be written as $\forall p[p(x)]$; consequently, it is identically false, and A^I is equivalent to

$$A(LP) \wedge \forall x \neg LP(x).$$

Example 3. If A is $LP(a)$, then A^I becomes $LP(a) \wedge \forall x \neg LP(x)$. This formula is equivalent to *false*—a counterpart of the fact that, in autoepistemic logic, LP has no stable expansions.

Now let $A(P, LP)$ be

$$A_0(P) \wedge \forall x(P(x) \supset LP(x)),$$

where $A_0(P)$ does not contain LP. The second conjunctive term here is similar to autoepistemic axiom (1). Then the result of introspective circumscription is the same as the result of McCarthy's "minimizing" circumscription applied to $A_0(P)$. More precisely:

Proposition 1. *If A is $A_0(P) \wedge \forall x(P(x) \supset LP(x))$, where $A_0(P)$ does not contain LP, then A^I is equivalent to*

$$\mathrm{Circum}(A_0(P); P) \wedge \forall x(LP(x) \equiv P(x)).$$

The expression $\mathrm{Circum}(A_0(P); P)$ denotes here the circumscription of P relative to $A_0(P)$, i.e., the conjunction of $A_0(P)$ and

$$\forall p[A_0(p) \wedge \forall x(p(x) \supset P(x)) \supset \forall x(p(x) \equiv P(x))].$$

(Proofs are given in the appendix.)

In Section 4 we show that some other forms of minimization can be reduced to introspective circumscription too.

Remark 1. Since this formalism includes quantifiers, it is interesting to compare it with the existing definitions of predicate autoepistemic logic, [Levesque, 1989] and [Konolige, 1987]. Our approach has some expressive capabilities not available in these two systems.

The first of them does not have equality (only "identity"), so that it cannot represent our Example 2. The second system does not permit "quantifying-in," so that it cannot represent the formula from Proposition 1.

Remark 2. Autoepistemic logic allows us to write the modal operator L in front of any formula, not necessarily atomic, while the use of the predicates LP_i corresponds to applying the belief operator to atoms only. This restriction is not very essential. In order to express that a nonatomic formula $F(x)$ is believed, we would introduce a new predicate constant P_{n+1}, explicitly defined by the axiom $P_{n+1}(x) \equiv F(x)$, and then write $LP_{n+1}(x)$.

3 Models of Introspective Circumscription

In this section we give a model-theoretic characterization of introspective circumscription.

3.1 Fixpoints

Let M be a model of $A(P, LP)$, where P and LP are tuples $P_1, \ldots, P_n$ and $LP_1, \ldots, LP_n$, as in Section 2. Consider the models of $A(P, LP)$ obtained from M by possibly changing the extents of some or all of the predicates P. For any $i = 1, \ldots, n$, let $\alpha_i(M)$ be the intersection of the extents of P_i in all such models. Furthermore, let $\beta(M)$ be the structure obtained from M by making the extent of each LP_i $(i = 1, \ldots, n)$ equal to $\alpha_i(M)$.

Proposition 2. *A model M of A satisfies A^I iff $\beta(M) = M$.*

We see that the models of introspective circumscription can be characterized as the fixpoints of the operator β. In this respect, introspective circumscription is similar to default logic and autoepistemic logic, where "extensions" and "stable expansions" are defined as the fixpoints of certain operators. The difference is that our fixpoints are *models*, rather than sets of formulas.[2]

The definitions of α_i and β can be reformulated using the following notation. Let $X_1, \ldots, X_n$ be sets of tuples of elements of M such that the length of each tuple in X_i equals the arity of P_i. We will write $M[X_1, \ldots, X_n/P_1, \ldots, P_n]$ for the structure which differs from M in that it interprets $P_1, \ldots, P_n$ as $X_1, \ldots, X_n$. Then

$$\alpha_i(M) = \bigcap_{\substack{X_1,\ldots,X_n: \\ M[X_1,\ldots,X_n/P_1,\ldots,P_n] \models A(P,LP)}} X_i, \qquad (3)$$

$$\beta(M) = M[\alpha_1(M), \ldots, \alpha_n(M)/LP_1, \ldots, LP_n].$$

Remark 3. The definition of β is closely related to the usual model-theoretic interpretation of modality. We can think of a model $M[X_1, \ldots, X_n/P_1, \ldots, P_n]$ as a "possible world," and then $\beta(M)$ interprets $LP_i(x)$ as the truth of $P_i(x)$ in each of these "worlds."

3.2 Propositional Case

The definition of introspective circumscription and its model-theoretic characterization given above are applicable, in particular, to the case when the arities of some or all of the predicates P_i equal 0, i.e., when some or all of the symbols P_i are propositional. In this case, X_i is a Boolean variable, and the intersection operation in (3) should be understood as conjunction.

If all symbols P_i are propositional, then a model of $A(P, LP)$ is simply a $2n$-tuple of truth values, representing the interpretations of $P_1, \ldots, P_n, LP_1, \ldots, LP_n$, and the operator β defined above is a mapping of $\{false, true\}^{2n}$ into itself. Instead of using β, we can describe the models of propositional introspective circumscription by means of a simpler operator γ, which maps $\{false, true\}^n$ into itself. This operator is defined as follows:

$$\gamma_i(Y) = \bigwedge_{X:A(X,Y)} X_i \qquad (i = 1, \ldots, n),$$

$$\gamma(Y) = (\gamma_1(Y), \ldots, \gamma_n(Y)).$$

Here both X and Y range over $\{false, true\}^n$, and X_i is the i-th component of X.

Proposition 3. *A model (X, Y) of a propositional formula $A(P, LP)$ satisfies $A^I(P, LP)$ iff $\gamma(Y) = Y$.*

In order to stress the fact that γ is determined A, we will sometimes write it as γ^A.

3.3 Examples

Example 4. Let A be $P_1 \wedge P_2$. Then

$$\gamma_1(Y_1, Y_2) = \bigwedge_{X_1,X_2:X_1 \wedge X_2} X_1 = true.$$

Similarly, $\gamma_2(Y_1, Y_2) = true$. Then

$$\gamma(Y_1, Y_2) = (true, true).$$

The only fixpoint of this operator is $(true, true)$. Proposition 3 shows that the models of A^I are the models of A in which both LP_1 and LP_2 are true. Consequently, the only model of A^I is

$$(true, true, true, true).$$

[2] McCarthy's circumscription is sometimes described as a "minimizing" and "model-theoretic" approach, while most other nonmonotinic formalisms are said to be "fixpoint" and "syntactic." Our analysis of introspective circumscription shows that there is no reason why these characteristic should be correlated.

Example 5. If A is $P_1 \vee P_2$, then

$$\gamma_1(Y_1, Y_2) = \bigwedge_{X_1, X_2 : X_1 \vee X_2} X_1 = \mathit{false}.$$

Similarly, $\gamma_2(Y_1, Y_2) = \mathit{false}$. Then

$$\gamma(Y_1, Y_2) = (\mathit{false}, \mathit{false}).$$

The only fixpoint of this operator is $(\mathit{false}, \mathit{false})$. The models of A^I are the models of A in which both LP_1 and LP_2 are false; there are 3 such models:

$$(\mathit{true}, \mathit{true}, \mathit{false}, \mathit{false}),$$
$$(\mathit{true}, \mathit{false}, \mathit{false}, \mathit{false}),$$
$$(\mathit{false}, \mathit{true}, \mathit{false}, \mathit{false}).$$

Example 6. If A is $LP_1 \vee P_2$, then

$$\gamma_1(Y_1, Y_2) = \bigwedge_{X_1, X_2 : Y_1 \vee X_2} X_1 = \mathit{false},$$

$$\gamma_2(Y_1, Y_2) = \bigwedge_{X_1, X_2 : Y_1 \vee X_2} X_2 = \neg Y_1,$$

$$\gamma(Y_1, Y_2) = (\mathit{false}, \neg Y_1).$$

The only fixpoint of this operator is $(\mathit{false}, \mathit{true})$. The models of A^I are the models of A in which LP_1 is false and LP_2 is true; there are 2 such models:

$$(\mathit{true}, \mathit{true}, \mathit{false}, \mathit{true}),$$
$$(\mathit{false}, \mathit{true}, \mathit{false}, \mathit{true}). \quad (4)$$

Example 7. Let A be $LP_1 \vee P_1$. Then

$$\gamma(Y_1) = \gamma_1(Y_1) = \bigwedge_{X_1 : Y_1 \vee X_1} X_1 = \neg Y_1.$$

It is clear that γ has no fixpoints. Consequently A^I is inconsistent.

4 Minimizing by Introspection

Our next goal is to extend Proposition 1 to other forms of minimization. We use the notation of [Lifschitz, 1985]. For any predicate symbols Q_1, Q_2, the expression $Q_1 \leq Q_2$ stands for $\forall x(Q_1(x) \supset Q_2(x))$, and $Q_1 = Q_2$ stands for $\forall x(Q_1(x) \equiv Q_2(x))$.

4.1 Parallel Circumscription

If P is a single predicate ($n = 1$), then, according to Proposition 1, minimizing P is expressed by the axiom $P \leq LP$. Consider the general case, when P is the list of n predicate constants $P_1, \ldots, P_n$. Several different minimizations can be applied then to $A_0(P)$. We can decide which of the predicates $P_1, \ldots, P_n$ will be minimized; let the minimized predicates be $P_1, \ldots, P_k$ ($1 \leq k \leq n$). Furthermore, about each of the remaining predicates we can decide whether it will be fixed or varied in the process of minimization; let the fixed predicates be $P_{k+1}, \ldots, P_l$ ($k \leq l \leq n$), and the remaining predicates $P_{l+1}, \ldots, P_n$ be varied. The result of the minimization corresponding to this "circumscription policy" is denoted by

$$\mathrm{Circum}(A_0(P); P_1, \ldots, P_k; P_{l+1}, \ldots, P_n). \quad (5)$$

How can this formula be expressed in terms of introspective circumscription?

It turns out that the introspective circumscription of

$$A_0(P) \wedge P_1 \leq LP_1 \wedge \ldots \wedge P_k \leq LP_k$$

corresponds to minimizing $P_1, \ldots, P_k$ with *all other predicates* $P_{k+1}, \ldots, P_n$ *varied* (i.e., to the case when $l = k$). Having the predicate P_i for some $i > k$ *fixed* can be expressed by $P_i = LP_i$, so that the effect of circumscription (5) can be achieved by applying introspective circumscription to

$$A_0(P) \wedge P_1 \leq LP_1 \wedge \ldots \wedge P_k \leq LP_k \wedge P_{k+1} = LP_{k+1} \wedge \ldots \wedge P_l = LP_l. \quad (6)$$

More precisely, we will prove the following generalization of Proposition 1:

Proposition 4. *If $A(P, LP)$ is (6), where $A_0(P)$ does not contain LP, then $A^I(P, LP)$ is equivalent to the conjunction of (5) with explicit definitions of the predicates LP (i.e., with formulas of the form $\forall x^i(LP_i(x^i) \equiv B_i(P, x^i))$, $i = 1, \ldots, n$, where $B_i(P, x^i)$ does not contain LP and has no parameters other than x^i).*

In other words, the introspective circumscription of (6) is a definitional, and consequently conservative, extension of (5).

Remark 4. The introspective circumscription A^I is completely determined by the formula A; it does not have any additional "policy" parameters. Different forms of minimization are represented by additional axioms: $P_i \leq LP_i$ minimizes P_i, and $P_i = LP_i$ makes it fixed. In this respect, introspective circumscription is similar to the formalisms proposed in [Lifschitz, 1987] and [Lifschitz, 1988]. There are some differences, however. First, this special case of introspective circumscription performs "global," rather than "pointwise" minimization. Second, additional axioms are required here to keep a predicate fixed, rather than to let it vary.

4.2 Prioritized Circumscription

Some applications require that *priorities* be established between the tasks of minimizing different predicates [McCarthy, 1986], [Lifschitz, 1985], or between the tasks of minimizing a predicate at different points [Lifschitz, 1987], [Lifschitz, 1988]. Our method can be easily extended to these more general forms of minimization.

Assume, for instance, that $n = 2$, and that we want to minimize P_1 and P_2 relative to $A_0(P_1, P_2)$, with P_1

given a higher priority. This assignment of priorities can be described using the lexicographical order $\preceq$ on pairs (p_1, p_2):

$$(p_1, p_2) \preceq (q_1, q_2) \equiv p_1 \leq q_1 \wedge (p_1 = q_1 \supset p_2 \leq q_2).$$

The corresponding prioritized circumscription is

$$\begin{aligned} &A_0(P_1, P_2) \\ &\quad \wedge \forall p_1 p_2 [A(p_1, p_2) \wedge (p_1, p_2) \preceq (P_1, P_2) \\ &\qquad \supset (p_1 = P_1 \wedge p_2 = P_2)]. \end{aligned} \tag{7}$$

In terms of introspective circumscription, minimizing P_1 at a higher priority and P_2 at a lower priority can be represented by the axioms $P_1 \leq LP_1$ and

$$P_1 = LP_1 \supset P_2 \leq LP_2.$$

The conjunction of these formulas can be written as $(P_1, P_2) \preceq (LP_1, LP_2)$. We will prove the following counterpart of Proposition 1 for prioritized circumscription:

Proposition 5. *If A is*

$$A_0(P_1, P_2) \wedge (P_1, P_2) \preceq (LP_1, LP_2),$$

where $A_0(P_1, P_2)$ does not contain LP_1, LP_2, then A^I is equivalent to the conjunction of (7) with the formulas $LP_1 = P_1$ and $LP_2 = P_2$.

"Chronological minimization" [Shoham, 1988] can be reduced to introspective circumscription, too. If P is a unary predicate whose argument is interpreted as time, then the axiom

$$\begin{aligned} \forall t \{\forall t' [earlier(t', t) \supset (P(t') \equiv LP(t'))] \wedge P(t) \\ \supset LP(t)\} \end{aligned}$$

expresses that a higher priority is given to minimizing P at earlier instants of time.

5 Introspection and Logic Programming

In [Gelfond, 1987], logic programs are translated into autoepistemic theories by inserting the modal operator L after each negation. We will define a semantics for logic programming on the basis of a similar translation, that replaces each negative literal $\neg P_i(\ldots)$ by $\neg LP_i(\ldots)$, and then applies introspective circumscription. Both transformation are based on the same idea of interpreting "negation as failure" in terms of introspection: A negative literal $\neg A$ in the body of a rule expresses that the program gives no grounds for believing in A.

First we will define the semantics for propositional programs and compare it with the "stable model" approach proposed in [Gelfond and Lifschitz, 1988] and with Gelfond's autoepistemic semantics. Then the general case will be discussed.

5.1 Propositional Programs

A *propositional logic program* is a finite set of formulas of the form

$$A_1 \wedge \ldots \wedge A_m \supset B, \tag{8}$$

where $m \geq 0$, each of $A_1, \ldots, A_m$ is a propositional literal (i.e., a propositional symbol possibly preceded by negation), and B a propositional symbol. These formulas are called the *rules* of Π; the antecedent of rule (8) is called its *body*, and the consequent B is its *head*.

Let Π be propositional logic program. By Π^L we denote the conjunction of the formulas obtained from the rules of Π by replacing each negative literal $\neg P_i$ in its body by $\neg LP_i$. We claim that Π^{LI}, the introspective circumscription of Π^L, represents the declarative meaning of Π. More precisely, we consider a propositional program Π "well-behaved" (meaningful) if, for each i, Π^{LI} entails exactly one of the literals LP_i, $\neg LP_i$. If a meaningful program Π implies LP_i, then we say that the value assigned by Π to P_i is *true*; otherwise, it is *false*.

Example 8. Consider the program Π consisting of just one rule:

$$\neg P_1 \supset P_2. \tag{9}$$

Then Π^L is $\neg LP_1 \supset P_2$, or, equivalently, $LP_1 \vee P_2$. In Example 6 we found that the introspective circumscription of this formul has two models. In both of them, the value of LP_1 is *false*, and the value of LP_2 is *true*. Consequently, Π is a well-behaved program, assigning the value *false* to P_1 and the value *true* to P_2. This conclusion agrees, of course, with the traditional procedural interpretation of (9).

Example 9. The program $\neg P_1 \supset P_1$ is not well-behaved: We showed in Example 7 that the introspective circumscription of $LP_1 \vee P_1$ is inconsistent.

The following proposition shows how this semantics can be reformulated in terms of the operator γ defined in Section 3.2.

Proposition 6. *(i) A propositional program Π is well-behaved iff γ^{Π^L} has a unique fixpoint. (ii) If a propositional program Π is well-behaved, and Y is the fixpoint of γ^{Π^L}, then the value assigned by Π to P_i equals Y_i.*

5.2 Stable Models

The basic definitions related to stable models [Gelfond and Lifschitz, 1988] can be stated as follows. Let Π be a propositional logic program whose propositional symbols are $P_1, \ldots, P_n$. For any $Y \in \{false, true\}^n$, let Π_Y be the program obtained from Π by deleting (i) each rule that has a negative literal $\neg P_i$ in its body such that $Y_i = true$, and (ii) all negative literals in the bodies of the remaining rules. Clearly, Π_Y is negation-free, so that it has a unique minimal Herbrand model.

This model is denoted by $S_\Pi(Y)$. Thus S_Π is a mapping of $\{false, true\}^n$ into itself. We say that Y is a *stable vector* of Π if it is a fixpoint of S_Π. Any stable vector of Π is a model of Π ([Gelfond and Lifschitz, 1988], Theorem 1), so that stable vectors can be also called *stable models*. According to the stable model semantics, a propositional program Π is meaningful if it has exactly one stable model; the value of P_i in that model is declared to be the value assigned to P_i by Π.

There is a simple relation between the construction used in the stable model semantics and the operator γ defined in Section 3.2:

Proposition 7. *For any propositional logic program* Π, $S_\Pi = \gamma^{\Pi^L}$.

Propositions 6 and 7 show that the semantics of propositional logic programs defined in Section 5.1 and the stable model semantics are exactly equivalent.

5.3 Autoepistemic Logic

According to Theorem 3 from [Gelfond and Lifschitz, 1988], the stable model semantics is also equivalent to the translation of logic programs into autoepistemic theories proposed in [Gelfond, 1987]. We conclude that autoepistemic logic, stable models and introspective circumscription provide three equivalent descriptions of the meaning of propositional logic programs.

This fact has an interesting consequence related to the more general problem of comparing autoepistemic logic with the propositional case of introspective circumscription. A propositional formula $A(P, LP)$ can be viewed as a formula of autoepistemic logic, if we identify an atom LP_i with the autoepistemic formula LP_i. What is the relation between the introspective circumscription A^I and the stable expansions of the autoepistemic theory with the axiom A?

We can answer this question in the special case when A has the form Π^L, where Π is a well-behaved program. According to Theorem 3 from [Gelfond and Lifschitz, 1988], such A has a single stable expansion. For every i, this stable expansion contains either LP_i or $\neg LP_i$. Our Propositions 6 and 7 and Theorem 3 from [Gelfond and Lifschitz, 1988] imply that such a literal belongs to the stable expansion of A if and only if it follows from A^I.

Michael Gelfond has proved a generalization of this fact to arbitrary propositional formula (personal communication).

5.4 General Programs

A *general logic program* is a finite set of formulas of form (8) where each of $A_1,\ldots,A_m$ is a literal, and B is an atom.

Both the autoepistemic approach and the stable model method reduce general programs to the propositional case by replacing every rule with variables by its ground instances. This amounts to ignoring any models of a logic program other than its Herbrand models.

When introspective circumscription is used, it is no longer necessary to use such a reduction. The definition of Π^L for general programs is similar to the definition for propositional programs given in Section 5.1: Π^L is the conjunction of the universal closures of the formulas obtained by replacing every negative literal $\neg P_i(\ldots)$ in the body of each rule of Π by $\neg LP_i(\ldots)$. As before, we claim that Π^{LI} represents the declarative meaning of Π. A general logic program Π assigns the value *true* or *false* to a ground atom $P_i(\ldots)$ depending on whether $LP_i(\ldots)$ or $\neg LP_i(\ldots)$ can be derived from Π^{LI} using Clark's equality axioms [Clark, 1978]. (For a logic program without functions, these axioms are simply the inequalities $a \neq b$ for all pairs of distinct constants a, b.)

The following example illustrates the difference between applying introspective circumscription directly to a program with variables, as suggested here, and replacing rules by their ground instances.

Example 10. Let Π be the program

$$\neg P_1(x) \supset P_2,$$
$$P_1(a).$$

Then Π^L is

$$\forall x(\neg LP_1(x) \supset P_2) \wedge P_1(a).$$

The introspective circumscription of this formula implies that LP_2 is equivalent to $\exists x(x \neq a)$; it entails neither LP_2 nor $\neg LP_2$. This absence of a definite result reflects the fact that, intuitively, Π assigns different values to P_2 depending on whether or not a is considered the only possible value of x. The answer would have been different if we replaced the first rule of Π by its only ground instance in this language,

$$\neg P_1(a) \supset P_2,$$

because the introspective circumscription of

$$(\neg LP_1(a) \supset P_2) \wedge P_1(a)$$

implies $\neg LP_2$.

6 Introspection and Commonsense Reasoning

We will illustrate the use of introspective circumscription for formalizing default reasoning on a simple example—Problem A1 from [Lifschitz, 1989]. We are given the following information: Blocks A and B are heavy; heavy blocks are normally located on the table; A is not on the table. The goal is to conclude that B is on the table.

In the appendix to [Lifschitz, 1989], this example is formalized in many different ways, using a number of nonmonotonic formalisms. Here are two more solutions, based on introspective circumscription.

The first formalization has the axioms

$$A \neq B, \quad heavy\ A, \quad heavy\ B, \quad \neg ontable\ A, \quad (10)$$

$$heavy\ x \wedge \neg ab\ x \supset ontable\ x, \quad (11)$$

and

$$ab\ x \supset L\_ab\ x.$$

According to Proposition 4 (Section 4.1), the introspective circumscription of (the universal closure of the conjunction of) these axioms has the same effect as minimizing ab relative to (10) and (11), with all predicates varied. Its result is

$$ab\ x = x = A,$$

and the desired conclusion

$$\neg ontable\ B \quad (12)$$

easily follows. This solution is essentially a "simple circumscriptive theory" in the sense of [McCarthy, 1986], expressed in the language of introspective circumscription.

The second possibility is to use axioms (10),

$$heavy\ x \wedge \neg L\_ab\ x \supset ontable\ x \quad (13)$$

and

$$ab\ A.$$

By simplifying the introspection condition for ab we get:

$$L\_ab\ x = x = A,$$

and this again implies (12). This solution is in the style of the autoepistemic approach of [Gelfond, 1989], like Solution 6 from [Lifschitz, 1989], except that we did not have to replace (13) by its ground instances.

Notice that both solutions given above allow us to prove

$$heavy\ x \wedge x \neq A \supset ontable\ x.$$

Like other solutions based on circumscription, they formalize "default reasoning in an open domain," which presents problems for Reiter's logic of defaults and for autoepistemic logic (see [Lifschitz, 1989], note to Problem A5).

This example is rather typical: Using introspective circumscription, we are able to model familiar applications of minimizing circumscription and autoepistemic logic to formalizing commonsense reasoning, and we get some additional benefits from the availability of quantifiers and equality. As another example of such benefits, we can observe that introspective circumscription easily handles "autoepistemic reasoning in an open domain": Given that block A is on the table, conclude that about any block other than A it is not known whether it is on the table ([Lifschitz, 1989], Problem E3). Here the only axiom is $ontable\ A$, and introspective circumscription gives

$$L\_ontable\ x \equiv x = A.$$

This is isomorphic to Example 1 from Section 2.2.

Acknowledgements

I would like to thank Kave Eshghi, Michael Gelfond, Matthew Ginsberg, Kurt Konolige, Hector Levesque, John McCarthy, Donald Perlis and Yoav Shoham for useful discussions related to the subject of this paper.

References

[Clark, 1978] Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Publishing Corporation, 1978.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Logic Programming: Proceedings of the Fifth International Conference and Symposium*, pages 1070–1080, 1988.

[Gelfond, 1987] Michael Gelfond. On stratified autoepistemic theories. In *Proc. AAAI-87*, pages 207–211, 1987.

[Gelfond, 1989] Michael Gelfond. Autoepistemic logic and formalization of commonsense reasoning: Preliminary report. In *Non-Monotonic Reasoning: 2nd International Workshop (Lecture Notes in Artificial Intelligence 346)*, pages 176–186. Springer-Verlag, 1989.

[Konolige, 1987] Kurt Konolige. On the relation between default logic and autoeistemic theories. *Artificial Intelligence*, 35(3):343–382, 1987.

[Levesque, 1989] Hector Levesque. All I know: A study in autoepistemic logic. *Artificial Intelligence*, 1989. To appear.

[Lifschitz, 1985] Vladimir Lifschitz. Computing circumscription. In *Proc. IJCAI-85*, pages 121–127, 1985.

[Lifschitz, 1987] Vladimir Lifschitz. Pointwise circumscription. In Matthew Ginsberg, editor, *Readings in Nonmonotonic Reasoning*, pages 179–193. Morgan Kaufmann, 1987.

[Lifschitz, 1988] Vladimir Lifschitz. Circumscriptive theories: a logic-based framework for knowledge representation. *Journal of Philosophical Logic*, 17:391–441, 1988.

[Lifschitz, 1989] Vladimir Lifschitz. Benchmark problems for formal nonmonotonic reasoning. In *Non-Monotonic Reasoning: 2nd International Workshop (Lecture Notes in Artificial Intelligence 346)*, pages 202–217, 1989.

[McCarthy, 1980] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1,2):29–39,171–172, 1980.

[McCarthy, 1986] John McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 26(3):89–118, 1986.

[Moore, 1985] Robert Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25(1):75–94, 1985.

[Morris, 1987] Paul Morris. Curing anomalous extensions. In *Proc. AAAI-87*, pages 437–442, 1987.

[Morris, 1989] Paul Morris. Autoepistemic stable closures and contradiction resolution. In *Non-Monotonic Reasoning: 2nd International Workshop (Lecture Notes in Artificial Intelligence 346)*, pages 60–73, 1989.

[Perlis, 1988] Donald Perlis. Autocircumscription. *Artificial Intelligence*, 36(2):223–236, 1988.

[Reiter, 1980] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1,2):81–132, 1980.

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

Appendix. Proofs of Theorems

Some of the proofs use the following fact:

Lemma 1. *$A^I(P, LP)$ implies $LP_i \le P_i$ $(1 \le i \le n)$.*

Proof. Assume $A^I(P, LP)$ and $LP_i(x)$. Then $A(P, LP)$ and $\forall p(A(p, LP) \supset p_i(x_i))$. Consequently, $P_i(x^i)$.

Proposition 1. *If A is $A_0(P) \wedge \forall x(P(x) \supset LP(x))$, where $A_0(P)$ does not contain LP, then A^I is equivalent to*

$$\mathrm{Circum}(A_0(P); P) \wedge \forall x(LP(x) \equiv P(x)).$$

Proof. If A is $A_0(P) \wedge P \le LP$, then A^I is

$$A_0(P) \wedge P \le LP \wedge \forall x[LP(x) \equiv \forall p(A_0(p) \wedge p \le LP \supset p(x))].$$

Lemma 1 shows that the second conjunctive term can be replaced by $LP = P$. Consequently, A^I can be written as

$$A_0(P) \wedge \forall x[P(x) \equiv \forall p(A_0(p) \wedge p \le P \supset p(x))] \wedge LP = P.$$

In this formula, the second conjunctive term can be replaced by the implication left-to-right

$$\forall x[P(x) \supset \forall p(A_0(p) \wedge p \le P \supset p(x))], \qquad (14)$$

because the implication right-to-left is trivial (substitute P for p). Furthermore, (14) is equivalent to

$$\forall p(A_0(p) \wedge p \le P \supset P \le p).$$

We conclude that A^I is equivalent to

$$A_0(P) \wedge \forall p(A_0(p) \wedge p \le P \supset P \le p) \wedge LP = P.$$

The conjunction of the first two terms is equivalent to $\mathrm{Circum}(A_0(P); P)$.

Proposition 2. *A model M of A satisfies A^I iff $\beta(M) = M$.*

Proof. Let M be a model of $A(P, LP)$. The condition $\beta(M) = M$ is equivalent to

$$M[\![LP_i]\!] = \alpha_i(M) \qquad (1 \le i \le n).$$

(By $M[\![Q]\!]$ we denote the extent of Q in M.) Furthermore, it is clear from the definition of α_i that

$$\alpha_i(M) = M[\![\lambda x \forall p(A(p, LP) \supset p_i(x_i))]\!].$$

Consequently, $\beta(M) = M$ iff

$$M[\![LP_i]\!] = M[\![\lambda x \forall p(A(p, LP) \supset p_i(x_i))]\!]$$

for all $i = 1, \ldots, n$, which is equivalent to saying that M satisfies the introspection conditions in (2).

Proposition 3. *A model (X, Y) of a propositional formula $A(P, LP)$ satisfies $A^I(P, LP)$ iff $\gamma(Y) = Y$.*

Proof. The introspection condition for a propositional symbol P_i is

$$LP_i \equiv \forall p(A(p, LP) \supset p_i).$$

Let (X, Y) be a model of $A(P, LP)$. It is clear from the definition of γ_i that $\gamma_i(Y)$ is the truth value of $\forall p(A(p, LP) \supset p_i)$ in this model. Consequently, $\gamma_i(Y) = Y_i$ iff (X, Y) satisfies the introspection condition for P_i, and $\gamma(Y) = Y$ iff (X, Y) satisfies all introspection conditions.

Proposition 4. *If $A(P, LP)$ is (6), where $A_0(P)$ does not contain LP, then $A^I(P, LP)$ is equivalent to the conjunction of (5) with explicit definitions of the predicates LP (i.e., with formulas of the form $\forall x^i(LP_i(x^i) \equiv B_i(P, x^i))$, $i = 1, \ldots, n$, where $B_i(P, x^i)$ does not contain LP and has no parameters other than x^i).*

Proof. To simplify notation, we will drop the superscript i in x^i. Using Lemma 1, we can transform A^I as follows:

$$\begin{aligned}
&A^I(P, LP) \equiv A^I(P, LP) \wedge \textstyle\bigwedge_{i=1}^n LP_i \le P_i \\
&\equiv A_0(P) \wedge \textstyle\bigwedge_{i=1}^k P_i \le LP_i \wedge \bigwedge_{i=k+1}^l P_i = LP_i \\
&\quad \wedge \textstyle\bigwedge_{i=1}^n \forall x\big[LP_i x \\
&\qquad \equiv \forall p\big(A_0(p) \wedge \textstyle\bigwedge_{j=1}^k p_j \le LP_j \wedge \bigwedge_{j=k+1}^l p_j = LP_j \\
&\qquad\qquad \supset p_i x\big)\big] \\
&\quad \wedge \textstyle\bigwedge_{i=1}^n LP_i \le P_i \\
&\equiv A_0(P) \wedge \textstyle\bigwedge_{i=1}^l P_i = LP_i \wedge \bigwedge_{i=l+1}^n LP_i \le P_i \\
&\quad \wedge \textstyle\bigwedge_{i=1}^n \forall x\big[LP_i x \\
&\qquad \equiv \forall p\big(A_0(p) \wedge \textstyle\bigwedge_{j=1}^k p_j \le P_j \wedge \bigwedge_{j=k+1}^l p_j = P_j \\
&\qquad\qquad \supset p_i x\big)\big] \\
&\equiv A_0(P) \wedge \textstyle\bigwedge_{i=1}^l P_i = LP_i \wedge \bigwedge_{i=1}^l \forall x(P_i x \equiv C_i(x)) \\
&\quad \wedge \textstyle\bigwedge_{i=l+1}^n [\forall x(LP_i x \equiv C_i(x)) \wedge LP_i \le P_i],
\end{aligned}$$

where $C_i(x)$ stands for

$$\forall p\big[\big(A_0(p) \wedge \bigwedge_{j=1}^{k} p_j \le P_j \wedge \bigwedge_{j=k+1}^{l} p_j = P_j\big) \supset p_i x\big]$$

$(i = 1, \ldots, n)$.

To simplify A^I further, we will first show that $A_0(P)$ implies $\forall x(C_i(x) \supset P_i x)$. Assume $C_i(x)$. Then

$$\big(A_0(P) \wedge \bigwedge_{j=1}^{k} P_j \le P_j \wedge \bigwedge_{j=k+1}^{l} P_j = P_j\big) \supset P_i x,$$

and consequently $P_i x$.

This fact allows us, first, to replace $\forall x(P_i x \equiv C_i(x))$ by $\forall x(P_i x \supset C_i(x))$, and, second, drop the term $LP_i \le P_i$. We conclude that $A^I(P, LP)$ is equivalent to

$$\begin{aligned} &A_0(P) \wedge \textstyle\bigwedge_{i=1}^{l} P_i = LP_i \\ &\quad \wedge \textstyle\bigwedge_{i=1}^{l} \forall x(P_i x \supset C_i(x)) \\ &\quad \wedge \textstyle\bigwedge_{i=l+1}^{n} \forall x(LP_i x \equiv C_i(x)). \end{aligned} \tag{15}$$

Define

$$B_i(P, x) = \begin{cases} P_i x, & \text{if } 1 \le i \le l; \\ C_i(x), & \text{if } l+1 \le i \le n. \end{cases}$$

Then (15) can be written as

$$\begin{aligned} &A_0(P) \wedge \textstyle\bigwedge_{i=1}^{l} \forall x(P_i x \supset C_i(x)) \\ &\quad \wedge \textstyle\bigwedge_{i=1}^{n} \forall x(LP_i x \equiv B_i(P, x)). \end{aligned}$$

Now take some $i \le l$, and consider the formula $\forall x(P_i x \supset C_i(x))$. It is equivalent to

$$\begin{aligned} &\forall x\big[P_i x \\ &\quad \supset \forall p\big((A_0(p) \wedge \textstyle\bigwedge_{j=1}^{k} p_j \le P_j \wedge \bigwedge_{j=k+1}^{l} p_j = P_j) \\ &\qquad \supset p_i x\big)\big], \end{aligned}$$

and consequently to

$$\forall p\big[\big(A_0(p) \wedge \bigwedge_{j=1}^{k} p_j \le P_j \wedge \bigwedge_{j=k+1}^{l} p_j = P_j\big) \supset P_i \le p_i\big].$$

For $i = k+1, \ldots, l$, the last formula is trivially true; for $i = 1, \ldots, k$, it is equivalent to

$$\begin{aligned} &\bar{\forall}\big[A_0(p_1, \ldots, P_{k+1}, \ldots, p_{l+1}, \ldots) \wedge \textstyle\bigwedge_{j=1}^{k} p_j \le P_j \\ &\qquad \supset P_i \le p_i\big]. \end{aligned}$$

where $\bar{\forall}$ stands for $\forall p_1 \ldots p_k p_{l+1} \ldots p_n$. Then

$$A_0(P) \wedge \bigwedge_{i=1}^{l} \forall x(P_i x \supset C_i(x))$$

is equivalent to

$$\begin{aligned} &A_0(P) \\ &\wedge \bar{\forall}\big[A_0(p_1, \ldots, P_{k+1}, \ldots, p_{l+1}, \ldots) \wedge \textstyle\bigwedge_{j=1}^{k} p_j \le P_j \\ &\qquad \supset \textstyle\bigwedge_{j=1}^{k} P_i \le p_i\big], \end{aligned}$$

which is equivalent to circumscription (5).

Proposition 5. *If A is*

$$A_0(P_1, P_2) \wedge (P_1, P_2) \preceq (LP_1, LP_2),$$

where $A_0(P_1, P_2)$ does not contain LP_1, LP_2, then A^I is equivalent to the conjunction of (7) with the formulas $LP_1 = P_1$ and $LP_2 = P_2$.

Proof. We write P for P_1, P_2 and p for p_1, p_2. As in the proof of Proposition 1, we use Lemma 1 to simplify A^I, as follows:

$$\begin{aligned} A^I &\equiv A_0(P) \wedge P \preceq LP \\ &\quad \wedge \forall x[LP_1(x) \equiv \forall p(A_0(p) \wedge p \preceq LP \supset p_1(x))] \\ &\quad \wedge \forall y[LP_2(y) \equiv \forall p(A_0(p) \wedge p \preceq LP \supset p_2(y))] \\ &\equiv A_0(P) \wedge P = LP \\ &\quad \wedge \forall x[P_1(x) \equiv \forall p(A_0(p) \wedge p \preceq LP \supset p_1(x))] \\ &\quad \wedge \forall y[P_2(y) \equiv \forall p(A_0(p) \wedge p \preceq LP \supset p_2(y))] \\ &\equiv A_0(P) \wedge P = LP \\ &\quad \wedge \forall x[P_1(x) \equiv \forall p(A_0(p) \wedge p \preceq P \supset p_1(x))] \\ &\quad \wedge \forall y[P_2(y) \equiv \forall p(A_0(p) \wedge p \preceq P \supset p_2(y))] \\ &\equiv A_0(P) \wedge P = LP \\ &\quad \wedge \forall x[P_1(x) \supset \forall p(A_0(p) \wedge p \preceq P \supset p_1(x))] \\ &\quad \wedge \forall y[P_2(y) \supset \forall p(A_0(p) \wedge p \preceq P \supset p_2(y))] \\ &\equiv A_0(P) \wedge P = LP \\ &\quad \wedge \forall p[A_0(p) \wedge p \preceq P \supset P_1 \le p_1] \\ &\quad \wedge \forall p[A_0(p) \wedge p \preceq P \supset P_2 \le p_2] \\ &\equiv A_0(P) \wedge P = LP \wedge \forall p[A_0(p) \wedge p \preceq P \supset P \le p] \\ &\equiv A_0(P) \wedge \forall p[A_0(p) \wedge p \preceq P \supset p = P] \wedge P = LP. \end{aligned}$$

It is convenient to prove first Proposition 7, and then Proposition 6.

Proposition 7. *For any propositional logic program Π, $S_\Pi = \gamma^{\Pi^L}$.*

Proof. Let $\Pi(P)$ be (the conjunction of the rules of) a propositional logic program, and let $\Pi^L(P, LP)$ be the result of replacing each literal $\neg P_i$ in $\Pi(P)$ by $\neg LP_i$, as defined in Section 5.1. Take any vector $Y \in \{false, true\}^n$. It is clear that the program Π_Y constructed as in Section 5.2 is equivalent to $\Pi^L(P, Y)$. It follows that a vector $X \in \{false, true\}^n$ satisfies $\Pi^L(X, Y)$ if and only if X is a model of Π_Y. Then the conjunction of X_i over all X satisfying $\Pi^L(X, Y)$ is the value of P_i in the minimal model of Π_Y. Consequently, $\gamma^\Pi(Y)$ is the minimal model of Π_Y, that is, $S_\Pi(Y)$.

The proof of Proposition 6 is based on the following lemma.

Lemma 2. *Let Π be a propositional logic program. If Y is a fixpoint of γ^{Π^L}, then (Y, Y) is a model of Π^{LI}.*

For instance, if Π is (9), then the second of the models (4) of Π^{LI} has the form (Y, Y), where Y is the fixpoint $(false, true)$ of γ^{Π^L}.

Proof. Let $\Pi(P)$ be (the conjunction of the rules of) a propositional logic program, let $\Pi^L(P, LP)$ be the result of replacing each literal $\neg P_i$ in $\Pi(P)$ by $\neg LP_i$,

and let Y be a fixpoint of γ^{Π^L}. In view of Proposition 7, Y is a stable vector of $\Pi(P)$. Consequently, Y is a model of $\Pi(P)$. But it is clear from the definition of Π^L that $\Pi(Y)$ is equivalent to $\Pi^L(Y,Y)$. Consequently, (Y,Y) is a model of $\Pi^L(P,LP)$.

Proposition 6. *(i) A propositional program* Π *is well-behaved iff* γ^{Π^L} *has a unique fixpoint. (ii) If a propositional program* Π *is well-behaved, and* Y *is the fixpoint of* γ^{Π^L}*, then the value assigned by* Π *to* P_i *equals* Y_i.

Proof. (i) Assume that Π is well-behaved, that is, for each i, Π^{LI} entails exactly one of the literals LP_i, $\neg LP_i$. Then Π^{LI} is consistent. It follows then from Proposition 3 that $\gamma^{P_i{}^L}$ has a fixpoint. If Y and Y' are two different fixpoints of $\gamma^{P_i{}^L}$, then, by Lemma 2, (Y,Y) and (Y',Y') are models of Π^{LI}, and they assign different values Y, Y' to LP. This contradiction shows that $\gamma^{P_i{}^L}$ has a unique fixpoint. Assume, on the other hand, that $\gamma^{P_i{}^L}$ has a unique fixpoint Y. Then, by Lemma 2, (Y,Y) is a model of Π^{LI}. Consequently, Π^{LI} is consistent, and it cannot entail both LP_i and $\neg LP_i$. If it entails neither LP_i nor $\neg LP_i$, then there exist models (X,Y) and (X',Y') of Π^{LI} such that $Y_i = \mathit{false}$ and $Y'_i = \mathit{true}$. Then, by Proposition 3, $\gamma^{P_i{}^L}$ has two different fixpoints Y, Y'. (ii) Assume that Π is well-behaved, and let Y be the only fixpoint of γ^{Π^L}. Then, by Lemma 2, (Y,Y) is a model of Π^L. Consequently, the value assigned to P_i by Π is Y_i.

Argument Systems:
a uniform basis for nonmonotonic reasoning

Fangzhen Lin*
Yoav Shoham
Department of Computer Science
Stanford University
Stanford, CA 94305

Abstract

We introduce the notion of *argument systems*. The key notions in an argument system are *inference rules, arguments, argument structures,* and *completeness conditions*. Argument structures are aggregations of arguments, which in turn are trees of inference rules. Completeness conditions define the closure of knowledge. From the informal standpoint, we find the definitions both simple and intuitive. From the formal standpoint, we show that default logic, autoepistemic logic, negation as failure principle, and circumscription are all special cases of the proposed framework.

1 Introduction

In this paper we propose a new framework for formalizing commonsense reasoning, and in particular its nonmonotonic nature. Many nonmonotonic logics exist in the literature. These include various forms of circumscription [McCarthy, 1986, Lifschitz, 1987], McDermott and Doyle's nonmonotonic logic I [McDermott and Doyle, 1980] and II [McDermott, 1982], Reiter's default logic [Reiter, 1980], Moore's autoepistemic logic [Moore, 1983], and Shoham's chronological ignorance [Shoham, 1988].

Our proposed framework, the notion of *argument systems*, is in a way a radical departure from previous approaches. The most novel feature of argument systems is that they are based entirely on *inference rules*. This contrasts with existing systems (such as the ones listed above), which are all *sentence-based* (default logic might be considered an exception). In fact, our proposal will not even presuppose that the underlying language has associated truth-functional semantics, although, of course, we allow that as a special case.

We find the framework advantageous in two respects. First, we show that default logic, autoepistemic logic, negation as failure, and circumscription are all special cases of the proposed framework. As an added benefit, we show that further insight can be gained into common features of the various special cases. We will thus be able to greatly extend previous unifying results, such as [Etherington, 1987, Konolige, 1987, and Shoham, 1988]. In particular, our unifying results suggest that a *generalized* "negation as failure" rule may be useful in implementing the existing nonmonotonic logics. Second, our framework is very simple, both mathematically and conceptually. In fact, it surprised us that such a simple framework can capture all those logics mentioned above, some of which are quite complex.

In this paper we stick to theoretical considerations, focussing on the expressive power of our framework. In a companion paper [Lin and Shoham, 1988] we discuss the applications of argument systems to "practical" cases, such as inheritance hierarchies. This paper is organized as follows. In Section 2 we define the theory of argument systems. In Section 3 we show that all major existing nonmonotonic logics are special cases of our framework. In Section 4 we summarize our results.

2 Basic Definitions

Throughout this section we fix a language $\mathcal{L}$ so that in the following when we say, for example, that φ is a well-formed formula (wff), we mean that φ is a wff in

*This work was supported by NSF grant IRI-8721701, IBM grant 1220042, and a gift from DEC Corporation. The first author acknowledges also the support of the Sloan Foundation.

$\mathcal{L}$. The only requirement we have of $\mathcal{L}$ be that it has a special operator "$\neg$" so that if φ is a wff, then $\neg\varphi$ is also a wff. At the moment no logical properties are assigned to the operator $\neg$. In particular, no inherent connection between φ and $\neg\neg\varphi$ is assumed. We do not even presuppose the existence of the notion of variables in $\mathcal{L}$, nor do we presuppose that it has associated truth-functional semantics. Its sole function is to provide us with a set of wffs.

Definition 2.1 *An* (inference) rule *is an expression of one of the following forms:*

1. *A, where A is a wff. A rule of this form will be called a* base fact.
2. *$A_1, \ldots, A_n \rightarrow B$, where $n > 0$, and A's and B are wffs. A rule of this form will be called* monotonic.
3. *$A_1, \ldots, A_n \Rightarrow B$, where $n > 0$, and A's and B are wffs. A rule of this form will be called* nonmonotonic.

Intuitively, a base fact represents our explicit knowledge about a specific domain. For example, the fact that *Tweety* is a bird can be represented as a base fact $bird(Tweety)$. A monotonic rule $A_1, \ldots, A_n \rightarrow B$ represents our deductive knowledge about the domain, it always enables us to conclude the B from the A's. For example, the knowledge "if a is a penguin, then it is also a bird" can be represented as the monotonic rule $penguin(a) \rightarrow bird(a)$. A nonmonotonic rule $A_1, \ldots, A_n \Rightarrow B$ represents our commonsense knowledge about the domain. It usually enables us to conclude the B from the A's, but there may be exceptions. For example, the statement "if a is a bird, then it flies" can be represented as the nonmonotonic rule $bird(a) \Rightarrow fly(a)$.

Chaining rules together into trees we get *arguments*, which are used to establish propositions.

Definition 2.2 *Let R be a set of rules. An argument in R is a rooted tree with labeled arcs, and is defined inductively as follows*

1. *If A is a base fact, then the tree consisting of A as a single node is an argument. A is the root of that argument.*
2. *If $p_1, \ldots, p_n$ are arguments whose roots are $A_1, \ldots, A_n$, respectively, and $A_1, \ldots, A_n \rightarrow B$ is a rule in R such that B is not a node in any one of the trees $p_1, \ldots, p_n$, then the tree p with B as its root and $p_1, \ldots, p_n$ as its immediate subtrees is an argument. p is said to be* formed *from $p_1, \ldots, p_n$ by using the monotonic rule $A_1, \ldots, A_n \rightarrow B$. This monotonic rule is the label of all links from the root B to its children.*
3. *If $p_1, \ldots, p_n$ are arguments whose roots are $A_1, \ldots, A_n$, respectively, and $A_1, \ldots, A_n \Rightarrow B$ is a rule in R such that B is not a node in any one of the trees $p_1, \ldots, p_n$, then the tree p with B as its root and $p_1, \ldots, p_n$ as its immediate subtrees is an argument. p is said to be* formed *from $p_1, \ldots, p_n$ by using the nonmonotonic rule $A_1, \ldots, A_n \Rightarrow B$. This nonmonotonic rule is the label of all links from the root B to its children.*

An argument p is said to be *for* φ, or φ is *supported* by p, if φ is the root of p. Notice that according to our definition of arguments, if the argument p is for φ, then φ can only appear as the root of p, that is, arguments are well founded. This is in order to prevent us from generating an infinite number of redundant arguments. For example, from the rules A and $A \rightarrow A$ we can only have one argument, that is, A. $A \rightarrow A$, $A \rightarrow A \rightarrow A$, etc. are not arguments.

Example 2.1 [Penguins do not fly] Let R be the set of following 8 rules:

r_1 : $True$

r_2 : $Penguin(a)$

r_3 : $Penguin(a) \rightarrow Bird(a)$

r_4 : $Penguin(a), \neg ab(penguin(a)) \rightarrow \neg Fly(a)$

r_5 : $Bird(a), \neg ab(bird(a)) \rightarrow Fly(a)$

r_6 : $Penguin(a) \rightarrow ab(bird(a))$

r_7 : $True \Rightarrow \neg ab(penguin(a))$

r_8 : $True \Rightarrow \neg ab(bird(a))$

There are also 8 arguments in R:

p_1 : $True$

p_2 : $Penguin(a)$

p_3 : $p_1 \stackrel{r_7}{\Rightarrow} \neg ab(penguin(a))$

p_4 : $p_1 \stackrel{r_8}{\Rightarrow} \neg ab(bird(a))$

p_5 : $p_2 \stackrel{r_5}{\rightarrow} Bird(a)$

p_6 : $p_2 \stackrel{r_6}{\rightarrow} ab(bird(a))$

p_7 : $p_3, p_2 \stackrel{r_4}{\rightarrow} \neg Fly(a)$

p_8 : $p_4, p_5 \stackrel{r_5}{\rightarrow} Fly(a)$

■

Grouping arguments together we get the notion of *argument structures*. These can be viewed as the sets of logically consistent arguments held by an agent.

Definition 2.3 *Let R be a set of rules. A set T of arguments in R is an* argument structure *of R if the following conditions are satisfied:*

1. *If p is a base fact in R, then $p \in T$.*
2. *T is* closed, *that is, for any $p \in T$ if p' is a subtree of p, then $p' \in T$.*
3. *T is* monotonically closed, *that is, if p is formed from $p_1, \ldots, p_n$ in T by a monotonic rule, then p is also in T.*
4. *T is* consistent, *that is, it does not contain arguments for both φ and $\neg\varphi$, for any wff φ.*

These conditions for argument structures are very weak. In particular, notice that no attempt is made to have argument structures be "maximal" in any sense. In fact, the related notion of *completeness* will play a central role in the following sections.

Intuitively, knowledge is complete if for every proposition about the world, either it or its negation is known. Clearly, complete knowledge is desirable but often unattainable. Nonmonotonic logics can be thought of as transforming partial knowledge to a more complete one.

In our formal system, we will define the notion of completeness with respect to a wff φ: an argument structure is complete about φ iff it contains an argument for either φ or its negation. We will then consider argument structures that are complete with respect to a class of wffs. It turns out that by being subtle about choosing these classes of wffs, we can capture both the similarities and differences between specific existing nonmonotonic logics. In practical, as we show in [Lin and Shoham, 1988], completeness conditions often enable us to pick out the intuitively right argument structures from the logically possible ones.

Another notion that will be used throughout this paper is the set of wffs *supported* by an argument structure.

Definition 2.4 *Let T be an argument structure. The set of wffs supported by T, written* $\mathrm{Wff}(T)$, *is defined as follows:*

$$\mathrm{Wff}(T) = \{\varphi \mid \textit{there is a } p \in T \textit{ such that } p \textit{ is for } \varphi\}$$

Thus an argument structure T supports a logically consistent set of beliefs, and T is complete about φ iff either $\varphi \in \mathrm{Wff}(T)$ or $\neg\varphi \in \mathrm{Wff}(T)$.

Example 2.2 [Penguins do not fly, continued] Assume the rules and arguments in Example 2.1. There are two argument structures of R:

$$T_1 = \{p_1, p_2, p_5, p_6\}$$
$$T_2 = \{p_1, p_2, p_5, p_6, p_3, p_7\}$$

Notice that only T_2 is complete about $ab(penguin(a))$ and $ab(bird(a))$. In fact, we have $\mathrm{Wff}(T_2) =$

$$\{True, Penguin(a), Bird(a), \neg ab(penguin(a)),$$
$$ab(bird(a)), \neg Fly(a)\}. \blacksquare$$

This concludes our basic definitions. To summarize, we start with the primitive notion of inference rules, and define those of arguments, argument structures, and completeness conditions. We shall call the resulting framework an *argument system*. In the following we show that this single framework is sufficient to capture the major existing nonmonotonic logics.

3 Capturing Existing Nonmonotonic Logics

In this section we show that the major existing nonmonotonic logics can be formulated naturally as special argument systems. The nonmonotonic logics considered here are default logic, autoepistemic logic, negation as failure principle, and circumscription.

3.1 Reiter's Default Logic

In this subsection, we suppose the underlying language is the ordinary first-order one, augmented with a second-order predicate ab. In the following, we shall use Reiter's notions without explanations [Reiter, 1980].

Let $\Delta = (W, D)$ be a closed default theory. Define $R(\Delta)$ to be the set of the following rules:

1. $True$ is a base fact.
2. If $A \in W$, then A is a base fact of $R(\Delta)$.
3. If $A_1, \ldots, A_n$, and B are first-order sentences and B is a consequence of $A_1, \ldots, A_n$ in first-order logic, then $A_1, \ldots, A_n \rightarrow B$ is a monotonic rule.[1]

[1] Some people may feel uneasy about this "rule schema." But we can replace it by any finite axiomatization of first-

4. If A is a first-order sentence, then $\neg A \rightarrow ab(A)$ is a monotonic rule.
5. If $A : MB_1, \ldots, MB_n/C$ is a default in D, then
$$A, \neg ab(B_1), \ldots, \neg ab(B_n) \rightarrow C$$
is a monotonic rule.
6. If B is a first-order sentence, then $True \Rightarrow \neg ab(B)$ is a nonmonotonic rule.

Definition 3.1 *An argument structure T of $R(\Delta)$ is* DL-complete *if for any first-order sentence A, either $ab(A)$ or $\neg ab(A)$ is in* Wff(T).

Theorem 1 *Let E be a consistent set of first-order sentences. E is an extension of Δ iff there is a DL-complete argument structure T of $R(\Delta)$ such that E is the restriction of* Wff(T) *to the set of first-order sentences.*

Proof: Let E be a consistent set of first-order sentences. According to Reiter's result (Theorem 1 in [Reiter 80]), E is an extension of Δ iff $E = E_0 \cup E_1 \cup \ldots \cup E_n \cup \ldots$, where E_i, $i \geq 0$, is defined as follows:

$$\begin{aligned} E_0 &= W \\ E_{i+1} &= Th(E_i) \cup \{C \mid A : MB_1, \ldots, MB_n/C \in D \\ &\quad \text{where } A \in E_i \text{ and } \neg B_1, \ldots, \neg B_n \notin E\} \end{aligned}$$

where $Th(E_i)$ is the closure of E_i under first-order consequence.

Suppose E is an extension of Δ, let $E' = E \cup \{ab(B) \mid \neg B \in E\} \cup \{\neg ab(B) \mid \neg B \notin E\}$. Obviously, for any first-order sentence B, either $ab(B)$ or $\neg ab(B)$ is in E', and E is the restriction of E' to the set of first-order sentences. We now prove that there is an argument structure T of $R(\Delta)$ such that $E' =$ Wff(T).

First we prove that for any $\varphi \in E'$ there is an argument for φ. To this end, we inductively prove that for any $i \geq 0$, if $\varphi \in E_i$, then there is an argument for φ. If $i = 0$, then $\varphi \in W$, thus φ is a base fact of $R(\Delta)$. Inductively, suppose we have done the proof for i. Let $\varphi \in E_{i+1}$, then φ is either in $Th(E_i)$ or there is a default $A : MB_1, \ldots, MB_n/\varphi$ in D such that $A \in E_i$. The former case is obvious. The later case is also easy to prove by using the inductive assumption and the fact that $A, \neg ab(B_1), \ldots, \neg ab(B_n) \rightarrow \varphi$ and $True \rightarrow \neg ab(B_k)$, $1 \leq k \leq n$, are rules. Therefore by Reiter's result, if $\varphi \in E$, then there is an argument for φ. From this by the definition of E', it is easy to see that if $\varphi \in E'$, then there is an argument for φ.

Now let T be the set of those arguments such that all wffs in them (as nodes) are in E'. Obviously $E' =$ Wff(T). It is easy to check that T is an argument structure of $R(\Delta)$.

Conversely, let T be a DL-complete argument structure of $R(\Delta)$ such that E is the restriction of Wff(T) to the set of first-order sentences. We prove that E is an extension of Δ, that is, $E = \cup E_i$. By induction on i, it is easy to see that $E_i \subseteq E$ for each i, that is, $\cup E_i \subseteq E$. Now let $\varphi \in E$, then there is an argument $p \in T$ for φ. By induction on the number of rules used in p, we prove that there is an $i \geq 0$ such that $\varphi \in E_i$. If p is a base fact, then $p \in W$, that is, $p \in E_0$. Inductively, suppose that for any proper subtree p' of p, if p' is for a first-order sentence, then the sentence is in $\cup E_i$. If p is formed by using first-order deduction rule, then trivially, $\varphi \in \cup E_i$. If p is formed by using the rule $A, \neg ab(B_1), \ldots, \neg ab(B_n) \rightarrow \varphi$, then by inductive assumption, there is an i such that $A \in E_i$. But $p \in T$, therefore $\neg ab(B_1), \ldots, \neg ab(B_n) \in$ Wff(T), so $B_1, \ldots, B_n \notin E$, thus $\varphi \in E_{i+1}$. This concludes the induction proof, thus we have proved that $E = \cup E_i$ and E is a default extension of Δ. ■

We notice the following two features of the transformation from default logic to argument systems. First, whereas default logic makes a distinction between meta-logic default rules and first-order logic, they are given an equal status in argument systems. Second, the notion of groundedness of default extensions corresponds to that of arguments in argument systems, and the notion of fixed-points in default logic corresponds to that of DL-completeness of argument structures.

It is interesting to notice that for normal default theories, we do not need the second-order predicate ab. Let $\Delta = (W, D)$ be a normal default theory, that is, every member of D has the form $A : MB/B$. Define $NR(\Delta)$ as the following set of rules:

1. If $A \in W$, then A is a base fact.
2. If $A_1, \ldots, A_n$, and B are first-order sentences, and B is a consequence of $A_1, \ldots, A_n$ in first-order logic, then $A_1, \ldots, A_n \rightarrow B$ is a monotonic rule.
3. If $A : MB/B$ is a default in D, then $A \Rightarrow B$ is a

order logic. We think it is neater to use a "rule schema" to capture first-order deduction than to introduce a bunch of axioms.

nonmonotonic rule.

We can prove the following theorem

Theorem 2 *A consistent set E of first-order sentences is a default extension of the default theory Δ iff there is a* maximal *argument structure T of $NR(\Delta)$ such that $E = \text{Wff}(T)$, where the notion of maximality is respect to the set inclusion of* $\text{Wff}(T)$.

Proof: For the proof of the theorem, it is enough to notice the direct correspondence between the normal default proofs in [Reiter 80] and the arguments in $NR(\Delta)$. ■

3.2 Autoepistemic Logic

In this subsection, let our language be the traditional proposition one augmented with a modal operator L. Throughout this section, let W be a set of wffs of the form:

$$LA \wedge \neg LB_1 \wedge \ldots \wedge \neg LB_n \supset C$$

where A, B's, and C are wffs without L. According to the results in [Konolige, 1987], it is enough to consider the AE-extensions of such W.

For any W, construct $R(W)$ as the set of following rules:

1. $True$ is a base fact.
2. If $LA \wedge \neg LB_1 \wedge \ldots \wedge \neg LB_n \supset C \in W$, then $LA, \neg LB_1, \ldots, \neg LB_n \rightarrow C$ is a monotonic rule.
3. If $A_1, \ldots, A_n \vdash B$, then $A_1, \ldots, A_n \rightarrow B$ is a monotonic rule, where A's and B are wffs without L, and $\vdash$ is the tautological consequence.
4. For any wff A without L, $A \rightarrow LA$ is a monotonic rule.
5. For any wff A without L, $True \Rightarrow \neg LA$ and $True \Rightarrow LA$ are nonmonotonic rules.

Definition 3.2 *An argument structure T of $R(W)$ is* AE-complete *if for any wff A without L, $\neg LA \in \text{Wff}(T)$ iff $A \notin \text{Wff}(T)$.* [2]

[2] Readers may notice that the definition of AE-completeness condition is a little different from the general definition of completeness conditions outlined in the last section. The trick here is that we are preventing from introducing an additional predicate. If we introduce a second-order predicate in and have the rule schemata $A \rightarrow in(A)$, $\neg in(A) \rightarrow \neg LA$, and $True \Rightarrow \neg in(A)$, then AE-completeness will be equivalent to AE'-completeness, where T is AE'-complete if for any wff A without L either LA or $\neg LA$ is in $\text{Wff}(T)$, and for any wff B without in either $in(B)$ or $\neg in(B)$ is in $\text{Wff}(T)$.

It is easy to see that if T is an AE-complete argument structure, then for any wff A without L, either LA or $\neg LA$ must be in $\text{Wff}(T)$. But the converse is generally not true. Notice that for any argument structure T of $R(W)$ it follows automatically that if $\neg LA \in \text{Wff}(T)$, then $A \notin \text{Wff}(T)$, because $A \rightarrow LA$ is a monotonic rule. Thus the AE-completeness condition means that if $A \notin \text{Wff}(T)$ and we have a choice of using either the rule $True \Rightarrow LA$ or the rule $True \Rightarrow \neg LA$ to expand the argument structure while maintaining consistency, then we can use the former one only when later on we can deduce A so that $A \in \text{Wff}(T)$.

Theorem 3 *A consistent stable set E is an AE-extension of W iff there exists an AE-complete argument structure T of $R(W)$ such that $E_0 = \text{Wff}(T)_0$, where E_0 ($\text{Wff}(T)_0$) is the restriction of E ($\text{Wff}(T)$) to the set of wffs without the operator L.*

Proof: In the proof of the theorem, we need the following result about AE-extension. It can be proved that a stable set E is an AE-extension of W iff

$$E_0 = \{\varphi \mid \varphi \text{ without } L \text{ and } W_E \vdash \varphi\} \qquad (1)$$

where $\vdash$ is tautological consequence and W_E is

$$W_E = \{C \mid LA \wedge \neg LB_1 \wedge \ldots \wedge \neg LB_n \supset C \in W, A \in E, \text{ and } B_1, \ldots, B_n \notin E\}$$

Return to the proof of the theorem, let E be an AE-extension of W. Define E' as follows:

$$E' = E_0 \cup \{LA \mid A \in E_0\} \cup \{\neg LA \mid A \notin E_0\}$$

Obviously, $E'_0 = E_0$ and for any A without L, if $A \notin E'$, then $\neg LA \in E'$. Therefore we only need to prove that there is an argument structure T of $R(W)$ such that $E' = \text{Wff}(T)$.

First we prove that for any $\varphi \in E'$, there is an argument p for φ such that any subtree (as an argument) of p is also for a wff in E'. According to the definition of E' and equation 1, it is enough to prove this for $\varphi \in W_E$. If $\varphi \in W_E$, then there is a wff $LA \wedge \neg LB_1 \wedge \ldots \wedge LB_n \supset \varphi$ in W such that $A \in E_0$ and $B_1, \ldots, B_n \notin E_0$. According to the definition of E', we have $LA \in E', \neg LB_1, \ldots, \neg LB_n \in E'$. Therefore using rules $True \Rightarrow LA$, $True \Rightarrow \neg LB_i$, $1 \leq i \leq n$, and

$LA, \neg LB_1, \ldots, \neg LB_n \rightarrow \varphi$ we get an argument for φ such that any subtree of the argument is also for a wff in E'.

Now let T be the set of those arguments such that they and their subtrees are all for wffs in E'. It is not hard to check that T is an argument structure of $R(W)$ and $E' = \text{Wff}(T)$.

Conversely, let T be an argument structure of $R(W)$ such that for any A without L, if $A \notin \text{Wff}(T)$, then $\neg LA \in \text{Wff}(T)$. Let E be the unique stable set such that $E_0 = \text{Wff}(T)_0$. We need to prove that E is an AE-extension of W, that is, equation 1 holds. Let us denote the right hand side of equation 1 by $Th(W_E)$.

It is easy to see that $W_E \subseteq \text{Wff}(T)_0$, thus $Th(W_E) \subseteq E_0$. Conversely, if $\varphi \in E_0$, then $\varphi \in \text{Wff}(T)$. By the induction on the numbers of rules used in the arguments for φ, we can show that $\varphi \in Th(W_E)$. For example, if an argument for φ is formed by using the rule $LA, \neg LB \rightarrow \varphi$, then $LA, \neg LB \in \text{Wff}(T)$ and $LA \wedge \neg LB \supset \varphi \in W$. Therefore $A \in \text{Wff}(T)$ and $B \notin \text{Wff}(T)$ (otherwise $\neg LA$ or LB will be in $\text{Wff}(T)$). Thus $\varphi \in W_E$. ■

Thus we have reformulated both default logic and autoepistemic logic in our framework of argument systems. It is not hard to see that the two reformulations are very similar under the correspondence $LA \leftrightarrow ab(\neg A)$ or $ab(A) \leftrightarrow L\neg A$. The main difference is that in the systems corresponding to autoepistemic theories, we have two kinds of nonmonotonic rules $True \Rightarrow LA$ and $True \Rightarrow \neg LA$, while in those corresponding to default theories, we have only one kind of nonmonotonic rules $True \Rightarrow \neg ab(A)$. Therefore we can expect that for autoepistemic theories which do not contain wffs of the form LA, the rule $True \Rightarrow LA$ will never be used and in this case autoepistemic extensions will coincide with default extensions. Indeed we have the following formal result.

Theorem 4 *Let W be an autoepistemic theory such that the elements of W have the form $\neg LB_1, \ldots, \neg LB_n \supset C$, where B's and C are wffs without L. Let $\Delta = (True, D)$ be the default theory such that D is the following set:*

$$\{:M\neg B_1, \ldots, M\neg B_n / C \mid \neg LB_1, \ldots, \neg LB_n \supset C \in W\}$$

For any stable set E, E is an AE-extension of W iff E_0 is a default extension of Δ.

Proof: Let $R'(W)$ be the subset of $R(W)$ without the nonmonotonic rules $True \Rightarrow LA$. Under the correspondence $LA \leftrightarrow ab(\neg A)$, T is an AE-complete argument structure of $R(W)$ if and only if T is an argument structure of $R'(W)$ such that for any wff A without L either LA or $\neg LA$ is in $\text{Wff}(T)$, and if T is a DL-complete argument structure of $R(\Delta)$. Therefore by Theorem 1 and Theorem 3, if E_0 is a default extension of Δ, then E is an AE-extension of W.

The converse is proved similarly by the correspondence $ab(A) \leftrightarrow L(\neg A)$ and the fact that if T is an argument structure of $R'(W)$ such that for any A without L, either LA or $\neg LA$ is in $\text{Wff}(T)$, then T is a DL-complete argument structure of $R(\Delta)$. ■

Autoepistemic (default) theories of the form in the theorem do not contain *prerequisites*. For theories that contain prerequisites, the above theorem is not true. Counterexamples can be found in [Konolige, 1987].

3.3 Negation as Failure

Negation as failure principle was proposed by Clark [Clark, 1978]. It is appealing for its efficiency. The principle can be captured nicely in our framework. Theoretically, it is enough to study the principle in a propositional language.

In the following, we first review a simplified version of Van Gelder's tight tree semantics [Van Gelder, 1986] for general logic programs with negation as failure. Then we propose a translation from logic programs to argument systems and connect them to Van Gelder's tight tree semantics.

An atom is a primitive proposition. A literal is an atom or the negation of an atom. A logic program with negation is a set of clauses with the following form:

$$A \leftarrow B_1, \ldots, B_n$$

where A is an atom and B's are literals.

Given a logic program W, Van Gelder associated it with two sets, SS and GF. Then the meaning of the program is that atoms in SS are true, atoms in GF are false, and atoms in neither SS nor GF are unclassified.

The sets SS and GF are defined through a collection of sets

$$SS_0, GF_0, RT_0, SS_1, GF_1, RT_1, \ldots$$

which we now describe.

We start with $SS_0 = GF_0 = \emptyset$ and RT_0 is the set of *tight NF-trees* of W. A tight NF-tree (negation as

failure derivation tree) of W is a tree whose nodes are literals such that a negated atom must be a leaf, any internal node must match the head of a rule in W and the sons of the internal node can be placed in a one-to-one correspondence with matching subgoals of the rule, and no node can have an identical ancestor.

For any ordinal k, define $(SS_{k+1}, GF_{k+1}, RT_{k+1})$ from (SS_k, GF_k, RT_k) as follows:

- Initialize SS_{k+1} to SS_k. For each tree in RT_k consisting of a single node, add that atom to SS_{k+1}.
- Initialize GF_{k+1} to GF_k. For each atom that matches the root of no tree in RT_k, add that atom to GF_{k+1}.
- Initialize RT_{k+1} to RT_k. In each tree in RT_{k+1}, delete all leaves that appear in SS_{k+1}, and delete all negated atoms (necessary leaves) that appear in GF_{k+1}. These nodes are considered proved and the trees "shrink" accordingly.

For each limit ordinal α, we define

$$SS_\alpha = \bigcup_{\beta<\alpha} SS_\beta, \quad GF_\alpha = \bigcup_{\beta<\alpha} GF_\beta, \quad RT_\alpha = \bigcap_{\beta<\alpha} RT_\beta$$

And finally we have

$$SS = \bigcup SS_\alpha \qquad GF = \bigcup GF_\alpha$$

Tight tree semantics can be captured nicely in our framework. For any program W, define $R(W)$ to be the set of following rules:

1. $True$ is a base fact.
2. For any clause $A \leftarrow B_1, \ldots, B_n$ in W, if $n = 0$ then A is a base fact, otherwise $B_1, \ldots, B_n \rightarrow A$ is a monotonic rule.
3. For any atom A, $True \Rightarrow \neg A$ is a nonmonotonic rule.

Definition 3.3 *An argument structure T of $R(W)$ is* NF-complete *if for any atom A, either A or $\neg A$ is in* Wff(T).

The following propositions verify the relationship between Van Gelder's tight tree semantics and the argument systems.

Lemma 3.1 *A tree p is a tight NF-tree of W iff p can be obtained from an argument in $R(W)$ by deleting the leaves $True$, which must be the son of a negated atom.*

Proof: Trivial from definitions. ■

Theorem 5 *For any NF-complete argument structure T of $R(W)$ and any atom A, if $A \in SS$, then $A \in$ Wff(T); if $A \in GF$, then $\neg A \in$ Wff(T).*

Proof: We prove by induction that for any ordinal α, if $A \in SS_\alpha$, then A is supported by T, and if $A \in GF_\alpha$, then $\neg A$ is supported by T. This is proved by using Lemma 3.1 and the fact that if an atom matches no tree in RT_α, then there is no argument for the atom, thus by the NF-completeness of T, the negation of the atom must be supported by T. ■

The converse to the theorem is not true according to our simplified definition of SS and GF. Whether or not it is true according to the full version of the tight tree semantics in [Van Gelder, 1986] is an open question.

A special class of logic programs, call *stratified* programs, has been studied intensively in the literature. Van Gelder [Van Gelder, 1986] proved that if W is a stratified program, then there is no unclassified atom, that is, for any atom A, either $A \in SS$ or $A \in GF$ (but not both). According to the above theorem, for stratified program W, if there is a NF-complete argument structure of $R(W)$, then it must be unique. The existence of the NF-complete argument structures can be proved by showing that the following set of arguments is one: $T = \{p \mid p$ is obtained from a tight NF-tree whose negated leaves are in GF by extending all its negated atoms to $True$ by a nonmonotonic rule$\}$.

It is interesting to notice the similarity between argument systems for default theories and argument systems for logic programs under the correspondence $\neg A \leftrightarrow \neg ab(\neg A)$, where A is an atom. Let W be a program. Let $R'(W)$ be the set of rules obtained from $R(W)$ by replacing every negated literal $\neg A$ by $\neg ab(\neg A)$ and adding the rule schema $A \rightarrow ab(\neg A)$, where A is an atom. Again we say that an argument structure T of $R'(W)$ is DL-complete if for any atom A, either $ab(\neg A)$ or $\neg ab(\neg A)$ is in Wff(T). We have the following proposition:

Proposition 3.1 *Let S be a set of atoms. Then there is an NF-complete argument structure T of $R(W)$ such that S is the set of atoms supported by T iff there is a DL-complete argument structure T' of $R'(W)$ such that S is the set of atoms supported by T'.*

Proof: Let T be a set of arguments in $R(W)$. Let T' be the set of trees obtained from T by replacing

every negated literal $\neg A$ by $\neg ab(\neg A)$. Then T' is a set of arguments in $R'(W)$ by appropriately changing the labels of the arcs. It is easy to see that T is an NF-complete argument structure of $R(W)$ iff the monotonic closure of T' is a DL-complete argument structure of $R'(W)$. ■

Therefore for any program W, the following default theory $\Delta = (P, D)$ captures the negation as failure rule:

1. If $A \leftarrow B_1, \ldots, B_n \in W$ does not contain negated atom, then

$$B_1 \wedge \ldots \wedge B_n \supset A \in P$$

(the implication behaves exactly the same as the monotonic rule $B_1, \ldots, B_n \rightarrow A$ in this case).

2. If $A \leftarrow B_1, \ldots, B_m, \neg B_{m+1}, \ldots, \neg B_n \in W$, then

$$B_1 \wedge \ldots \wedge B_m : M\neg B_{m+1}, \ldots, M\neg B_n / A \in D$$

It is interesting to notice the difference between the transformation in [Gelfond, 1987] from logic programs to autoepistemic theories and the transformation here from logic programs to default theories. Under Gelfond's transformation, a clause

$$A \leftarrow B_1, \ldots, B_m, \neg B_{m+1}, \ldots, \neg B_n$$

will be transformed into the sentence:

$$B_1 \wedge \ldots \wedge B_m \wedge \neg LB_{m+1} \wedge \ldots \wedge \neg LB_n \supset A$$

which is equivalent to the default:

$$: M\neg B_{m+1}, \ldots, M\neg B_n / (A \vee \neg B_1 \vee \ldots \vee \neg B_m)$$

according to Theorem 4. Incidently these two transformations lead to the same result for stratified programs. For non-stratified programs, these two transformations may lead to different results. In those cases, as we have shown above, our transformation seems closer to some procedural semantics for logic programs.

3.4 Circumscription

Finally we come to circumscription, one of the best well-known formalisms for nonmonotonic reasoning. In a trivial sense, circumscription can be reformulated in our framework because in all cases, the result of circumscription is explicitly represented as a second-order sentence. What we need to do is just let the second-order sentence as the base fact and have a class of monotonic rules corresponding to second-order deduction. Of course, no one will find this reformulation interesting because it sheds no further lights on either circumscription or argument systems. Intuitively, any meaningful reformulation of circumscription should only start at the original first-order theory and come up with the result of circumscription by some monotonic and nonmonotonic rules. In this subsections we propose a such reformulation.

Throughout this subsection, Let $\mathcal{L}$ be a first-order language with equality. We consider the circumscription of P in W with Z allowed to vary, where W is a sentence, and, for the sake of simplicity, P and Z are two different unary predicates. All of the following results can be extended to the cases where P and Z are two tuples of predicates straightforwardly.

Semantically, circumscribing P in W with the predicate Z allowed to vary corresponds to consider the sentences that are true in all minimal models of W according to the relation $<_P^Z$ which is defined as follows. Let M_1 and M_2 be two models satisfying W. Defining $M_1 <_P^Z M_2$ iff $\mathcal{L}$ can be expanded to $\mathcal{L}'$, M_1 can be extended to M_1' as a structure of $\mathcal{L}'$, and there are two predicates $B(x)$ and $D(x)$ in $\mathcal{L}'$ such that

1. For any sentence A in $\mathcal{L}$, $M_2 \models A$ iff $M_1' \models A(P/B)(Z/D)$, where $A(P/B)(Z/D)$ is the result of replacing $P(x)$ and $Z(x)$ everywhere in A by $B(x)$ and $D(x)$, respectively.
2. $M_1' \models \forall x(P(x) \supset B(x)) \wedge \neg\forall x(B(x) \supset P(x))$.

Condition 1 means that M_1 and M_2 interpret all the predicates except P and Z in the same way, and M_2 interprets P and Z in the same way that M_1' does for B and D, respectively. Condition 2 means that P is "less than" B in M_1', therefore the denotation of P in M_1 is "less than" that in M_2. We call B and D the *minimizing predicates* in $\mathcal{L}'$ under M_1'.

Thus M is a minimal model of W (w.r.t. $<_P^Z$) if M is a model of W and there is not a model M' of W such that $M' <_P^Z M$.

Readers may notice that our definition of "less than" ($<_P^Z$) is different than the traditional one such as that in [Lifschitz, 1985]. According the traditional definition, M_1 is "less than" M_2, written $M_1 <_P^{'Z} M_2$, if

1. the domain of M_1 is the same as that of M_2.
2. $M_1(F) = M_2(F)$ for any symbol F different than P and Z.

3. $M_1(P) \subset M_2(P)$.

It is easy to see that if $M_1 <'^Z_P M_2$, then $M_1 <^Z_P M_2$. Conversely, if $M_1 <^Z_P M_2$, then there is a structure M_1' such that for any $\mathcal{L}$-sentence A, $M_2 \models A$ iff $M_1' \models A$, and $M_1 <'^Z_P M_2$. Therefore for the minimal models that can be captured by first-order theories, it does not matter whether the minimality is according to $<^Z_P$ or $<'^Z_P$. The basic merit of our definition is that it enables us to compare two structures even when they do not have the same domain.

In order to capture circumscription in argument systems, we need to expand the language $\mathcal{L}$ to $\mathcal{L}_\infty$ by adding two new unary predicates Q and R. Corresponding to circumscribe P in W with Z allowed to vary, we have the following set of rules, written $R(W, P; Z)$:

1. W is a base fact.
2. $W(P/Q)(Z/R)$ is a base fact.
3. If $A_1, \ldots, A_n \vdash B$, then $A_1, \ldots, A_n \rightarrow B$ is a monotonic rule, where $\vdash$ is the first-order consequence and A's and B are sentences in $\mathcal{L}_\infty$.
4. $\forall x(P(x) \supset Q(x))$ is a base fact.
5. $\neg\forall x(Q(x) \supset P(x)) \rightarrow \mathit{False}$ is a monotonic rule, where False is the always false proposition.
6. If A is a sentence in $\mathcal{L}_\infty$ that contains nither Z nor P, then $\mathit{True} \Rightarrow A$ is a nonmonotonic rule.

Definition 3.4 *Let T be an argument structure of $R(W, P; Z)$. T is* CF-complete *if for any sentence A in $\mathcal{L}_\infty$ that contains predicates only from P and Q, either $A \in \mathrm{Wff}(T)$ or $\neg A \in \mathrm{Wff}(T)$.*

It is easy to see that if T is a CF-complete argument structure, then $\forall x(P(x) \equiv Q(x))$ must be supported by T. For otherwise $\neg\forall x(P(x) \equiv Q(x))$ must be supported. This means that $\neg\forall x(Q(x) \supset P(x))$ must be supported. But that is impossible for then False would be supported, and T would not be consistent. Actually this is the only property of CF-completeness that we will use in the proofs of the following results.

The following lemma means that if $M_1 <^Z_P M_2$ and M_2 satisfies every sentence in the argument p of $R(W, P; Z)$, then M_1 also satisfies every sentence in p. Therefore $R(W, P; Z)$ always generates "sound" arguments according to $<^Z_P$.

Lemma 3.2 *Let M_1 and M_2 be models of W in $\mathcal{L}$, M_2' an extension of M_2 in $\mathcal{L}'$, and $M_1 <^Z_P M_2$ with the minimizing predicates B and D in $\mathcal{L}'$ under M_1' such that*

1. *$\mathcal{L}_\infty$ is a subset of $\mathcal{L}'$.*
2. *B, D, Q, and R are different.*
3. *$M_1'(Q) = M_1'(B)$, $M_1'(R) = M_1'(D)$, $M_2'(Q) = M_2(P)$, and $M_2'(R) = M_2(Z)$, where $M(X)$ is the denotation of X in M.*

Then for any argument p in $R(W, P; Z)$, if $M_2' \models p$, that is, for every node φ in p, $M_2' \models \varphi$, then also $M_1' \models p$.

Proof: The lemma can be proved by induction on the structure of p and the following two facts:

1. For any sentence A that contains neither Z nor P, $M_2' \models A$ iff $M_2 \models A(Q/P)(R/Z)$ iff $M_1' \models A(Q/B)(R/D)$ iff $M_1' \models A$, where $F(X_1/Y_1)(X_2/Y_2)$ is the result of replacing X_1 and X_2 everywhere in the formula F by Y_1 and Y_2, repectively.
2. $M_1' \models \forall x(P(x) \supset Q(x))$ iff $M_1' \models \forall x(P(x) \supset B(x))$.

■

Theorem 6 *For any CF-complete argument structure T of $R(W, P; Z)$, there is a minimal model of W such that it satisfies every $\mathcal{L}$-sentence of* Wff(T). *Conversely, if M is a minimal model of W, then there is a CF-complete argument structure T of $R(W, P; Z)$ such that M satisfies every $\mathcal{L}$-sentence of* Wff(T).

Proof: Let T be a CF-complete argument structure. Then Wff(T) must be a consistent set of first-order sentences. Thus there is a structure M in $\mathcal{L}_\infty$ satisfying Wff(T). By the CF-completeness of T, $\forall x(P(x) \equiv Q(x))$ must be supported. Thus M interprets P and Q in the same way. Because $W(P/Q)(Z/R) \in \mathrm{Wff}(T)$, we can assume that M also interprets Z and R in the same way. We assert that the restriction of M to $\mathcal{L}$, written M_2, must be minimal. Suppose otherwise, then there is a model M_1 of W such that $M_1 <^Z_P M_2$ with minimizing predicates B and D in $\mathcal{L}'$ under M_1'. It is easy to see that we can extend M_2 to M_2', and rename B and D if necessary so that the conditions in Lemma 3.2 are satisfied. Then M_2' is an extension of M. Therefore M_2' satisfies Wff(T). By Lemma 3.2, M_1' must also satisfy Wff(T). But this is impossible by the CF-completeness of T.

For then $M_1' \models \forall x(P(x) \equiv Q(x))$, which is equivalent to $M_1' \models \forall x(P(x) \equiv B(x))$, a contradition with $M_1 <_P^Z M_2$. Therefore M' is a minimal model of W and satisfies Wff(T).

Conversely let M is a minimal model of W in $\mathcal{L}$. We can construct a CF-complete argument structure T as follows. Let S be the set of $\mathcal{L}_\infty$ sentences that contains neither P nor Z, and are true in M when Q and R are replaced everywhere by P and Z, respectively. Then all the sentences in S can be assumed to be true by default. We assert that the monotonic closure of the set of arguments generated by making these default assumptions must be a CF-complete argument structure. First T is consistent, that is, $S' = \{W, W(P/Q)(Z/R), \forall x(P(x) \supset Q(x))\} \cup S$ is a consistent set of first-order sentences and does not imply $\neg\forall x(Q(x) \supset P(x))$ in first-order logic. S' is consistent is easy to see because the extension of M by assigning P and Q, and Z and R the same denotations is a model of S'. We prove that S' does not imply $\neg\forall x(Q(x) \supset P(x))$ by proving that in fact, sentence $\forall x(Q(x) \supset P(x))$ must be a logical consequence of S'. For otherwise there is a structure M' which satisfies $S' \cup \{\neg\forall x(Q(x) \supset P(x))\}$. Let M_1 be the restriction of M' to $\mathcal{L}$. Then $M_1 <_P^Z M$ with the minimizing predicates Q and R in $\mathcal{L}_\infty$ under M'. Therefore $\forall x(P(x) \equiv Q(x))$ is a logical consequence of S'. By the definition of S', for any sentence A that contains only Q, either $A \in S'$ or $\neg A \in S'$. Thus T must be CF-complete. Obviously M satisfies every $\mathcal{L}$-sentence in Wff(T). ■

Corollary 6.1 *For any $\mathcal{L}$-sentence A, A is true in all minimal models of W w.r.t. $<_P^Z$ iff $\neg A$ is not supported by any CF-complete argument structures of $R(W, P; Z)$.*

4 Some General Remarks

We have defined a theory of argument systems that, although simple, is sufficient to capture the major existing nonmonotonic logics. The main concepts are those of inference rules, arguments, argument structurs, and completeness conditions. We noted that the major existing nonmonotonic logics can be viewed as particular choices made about the underlying sets of rules and the completeness conditions.

Notice that the argument systems used in simulating the exisiting nonmonotonic logics have the following two features in common:

1. All the nonmonotonic rules in the systems are of the form: $True \Rightarrow \varphi$. This means that the set of wffs supported by an argument structure can always be represented as the closure of a subset of $\{\varphi \mid True \Rightarrow \varphi$ is a nonmonotonic rule$\}$ under monotonic rules, that is, all of the nonmonotonic logics are *presumptive*. This can be considered a support for Poole's *Theorist* [Poole, 1986] which tries to formalize commonsense reasoning as hypothesis formation process.

2. All the systems have some completeness conditions which require that for a class of wffs either they or their negations must be supported by an argument structure. This suggests that a generalized "negation as failure" rule might be useful in the implementing of these nonmonotonic logics. For example, for default logic, an argument structure must be complete about the wffs of the form $ab(A)$. Therefore if we allow second-order predicates in logic programs, then default logic can be simulated by applying the negation as failure rule to the abnormality predicate.

Some interesting results follows from these reformulations. We have shown a new result relating default logic and autoepistemic logic, and provided a new way to capture negation as failure rule in default logic. We believe that more results about the relationships among the major existing nonmonotonic logic can be obtained by studying these reformulations.

It is also worth to notice that all of the argument systems except those corresponding to logic programs have a class of monotonic rules that captures classical logic. It may be very interesting to see what happen if we weaken this class of rules, for example, by restricting ourself on a special class of sentences or a decidable segment of first-order logic.

References

[Clark, 1978] Keith Clark, Negation as Failure, in *Logics and Databases*, eds. by Gallaire and Minker, Plenum Press, 293–322, 1978.

[Etherington, 1987] David Etherington, Relating default logic and circumscription, *IJCAI—87*.

[Gelfond, 1987] Michael Gelfond, On Stratified Autoepistemic Theories, *AAAI-87*, 207–211.

[Konolige, 1987] Kurt Konolige, On the Relation Between Default Logic and Autoepistemic Logic, In *Readings in Nonmonotonic Reasoning*, edited by Matthew Ginsberg, Morgan Kaufmann, 1987.

[Lifschitz, 1985] Vladimir Lifschitz, Computing Circumscription, in *IJCAI-85*, 121–127.

[Lifschitz, 1987] Vladimir Lifschitz, Pointwise Circumscription, in *Readings in Nonmonotonic Reasoning*, Morgan Kaufmann, 1987, ed. by M. Ginsberg.

[Lin and Shoham, 1988] Fangzhen Lin and Yoav Shoham, Applications of argument systems, Draft, 1988. Submitted to *IJCAI - 89* (The draft and the paper have been put together into the following Technical Report).

[Lin and Shoham, 1988a] Fangzhen Lin and Yoav Shoham, Argument systems: a uniform basis for nonmonotonic reasoning, Technical Report CS-1243, 1988, Department of Computer Science, Stanford University.

[McCarthy, 1986] John McCarthy, Applications of Circumscription to Formalizing Commonsense Knowledge, *Artificial Intelligence* 28 (1986), 89–118.

[McDermott, 1982] Drew V. McDermott, Nonmonotonic Logic II, *JACM*, 29/1 (1982), 33–57.

[McDermott and Doyle, 1980] Drew V. McDermott and Jon Doyle, Nonmonotonic Logic I, *Artificial Intelligence*, 13(1980), 41–72.

[Moore, 1983] Robert Moore, Semantical considerations on Nonmonotonic Logic, *Conference Proceedings of IJCAI-83.*

[Poole, 1986] David Poole, THEORIST: A Logical Reasoning System for Defaults and Diagnosis, (1986.) University of Waterloo. Department of Computer Science. (WATECS) CS-86-06.

[Reiter, 1980] Raymond Reiter, A Logic For Default Reasoning, *Artificial Intelligence* 13(1980) 81–132.

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

[Van Gelder, 1986] Alan Van Gelder, Negation as Failure Using Tight Derivations for General Logic Programs, *Proceedings of IEEE Conference on Logic Programming*, 1986, 127–138.

Analogical Reasoning, Defeasible Reasoning, and the Reference Class

R. P. Loui *
Dept. of Computer Science and
Dept. of Philosophy
Washington University
St. Louis, MO 63130
loui@ai.wustl.edu

Abstract

This paper attempts four things. It demonstrates the possibility of accounting for Russell-style and Clark-style analogical reasoning in an existing framework for statistical reasoning. It critically reviews the proposals made by Clark for defeasible analogical reasoning and shows how they can be understood better simply as defeasible reasoning. It argues that generalization from the single case is not as desirable as projection from the single case; the difference has to do with the defeasibility of the inference. Finally, it muses about the prospects for an appropriate control strategy for statistical reasoning limited to a small number of cases.

1 Introduction.

1.1 The Logical Problem of Analogy.

Analogy has been studied in a variety of activities by AI authors, notably in problem solving, in learning, and in common-sense reasoning (consider, for instance, [Winston80], [Kedar-Cabelli86], and [Gentner83]). An excellent recent review of this work can be found in Stuart Russell's thesis [Russell87].

Most of the hopes to use analogy are as ampliative, unsound inference, but inference we are nevertheless willing to perform because of epistemological constraints – we don't know enough to do full induction. In the present treatment of analogy, I am equally concerned with situations in which our willingness to perform analogical reasoning derives from computational constraints. We may not want to include in our inductive reasoning as many cases as we are capable of recalling and considering. In the extreme case, we may even want to reason from a single case. It is appropriate to consider such reasoning from a single case to be analogy.

Among the many problems that seem to involve analogy, Russell distinguished the logical problem of analogy. This is the aspect of analogy of most interest here. How is analogical reasoning justified? Presumably, it is some form of induction, in the presence of added assumptions. A property is transferred from source case to target case in virtue of some shared properties. Apparently, there is inductive support for the possession of that property given the similarity. The interesting challenge is to render those assumptions in some underlying reasoning framework in such a way that the single source case contributes in the inference: it cannot be just that the probability of the transferred property given the shared properties is known to be high, because that makes the source case redundant.

Russell's approach was to justify analogical inference in a deductive framework. My approach will be to cast analogy as a special case of a logically sophisticated kind of statistical reasoning. Russell had examined as candidates for the explication of analogy less expressive frameworks for induction than the one considered here. Part of this exercise is to relieve his pessimism about the prospects of an inductive explication of analogy. We should welcome this relief because we all continue to think of analogy as inductive. Analogical reasoning so closely resembles inductive reasoning that it is a travesty to suggest that there can be no meaningful reduction. This paper hopes to correct any misapprehensions about the relation between inductive (or statistical) reasoning and analogical reasoning.

More importantly, basing analogy on statistical inference allows the capture of a wider class of common-sense reasoning. The study of reasoning from a small number of cases is the natural extension of analogy with a single case. Philosophers of induction may not see the point of reasoning from a small number of cases, just as they have often turned their noses to the special case of analogy. They might think just to run the induction mill on the cases available, whether they be one, three, or a thousand. but here's the point: we are sometimes unwilling to complete this computation. Our control strategy for arriving at interesting partial computations must be informed by what we know about the legitimacy of inferences from a small number of cases.

A different example of common-sense reasoning that extends analogical reasoning is studied by Peter Clark [Clark88]. Here, the problem is not with the multiplicity of cases, but with the multiplicity of shared properties. Clark's inferences are interesting for a variety of reasons, one of which is that it uses analogy to amplify defeasible reasoning. It is Clark's ambitions that most clearly point out the need to base analogy in an appropriate statistical framework. What seems to be a complex use of analogy turns out to be simply the use of specificity in defeasible statistical reasoning; what seems to be an unquestionably sound maxim for transferring from source to target turns out to be improper in the most disturbing and unexpected statistical way.

*I have had useful discussions with Stuart Russell, Ben Grosof, Guillermo Simari, Josh Tenenberg, Henry Kyburg, John Pollock, Mike Wellman, Fahiem Bacchus, and Dana Nau on this subject in that temporal order. Thanks to Michael Anderson for leaving Stuart's thesis on the coffee table.

1.2 Russell and Clark on Analogical Inference.

Russell's deductive account [Russell87] of analogical reasoning is at once authoritative and demonstrating exemplary clarity. Peter Clark's recent attempt to generalize the analogical reasoning paradigm to arbitrate among conflicting defeasible arguments is refreshing and exciting [Clark88]. These two papers represent two of the most formal accounts of analogical reasoning.

Russell is interested in inferences such as

Nationality determines Language
Language(Louis, French)

Nationality(Louis, France)

Nationality(Antoinette, France)
thus,
Language(Antoinette, French)

by the analogy of the target, Antoinette, to the source, Louis. Louis's language, namely French, determines Antoinette's language to be French, in virtue of their important similarity. Russell justified this reasoning by taking the determination relation to mean, roughly,

(w)(y)(z).
(Nationality(w, y) ∧ Language(w, z)) →
(x). Nationality(x, y) → Language(x, z).

This makes the inference valid in a first order system.

Peter Clark is interested in inferences such as

Over-Block(x) INDICATES ¬Sand-At(x)
Sand-Nearby(x) INDICATES Sand-At(x)
Late-Fault(x) INDICATES Sand-At(x)
Unfavorable-Environment(x) INDICATES ¬Sand-At(x)

Over-Block($well_1$)
Sand-Nearby($well_1$)
Late-Fault($well_1$)
Unfavorable-Environment($well_1$)
Sand-At($well_1$)

Over-Block($well_2$)
Sand-Nearby($well_2$)
Late-Fault($well_2$)
Unfavorable-Environment($well_2$)
thus,
Sand-At($well_2$)

by the analogy of the target, $well_2$, to the source, $well_1$.

The resolution of the conflicting arguments in the case of $well_1$ determines the resolution of the conflicting arguments in the case of $well_2$. I will say that Sand-Nearby and Late-Fault are factors that indicate the ultimate conclusion. Over-Block and Unfavorable-Environment are factors that counter-indicate the ultimate conclusion. Note that in Russell's notation, we might have said that Over-Block(x, true) determines Sand-At(x, false), if Over-Block had been the lone indicating factor. The easiest way to translate between Russell and Clark is to consider the case wherein Russell's relations are bivalent in the second argument (and determinations apply only when values are true).

Clark is interested in an additional behavior that I will call "monotonicity of conflict resolution." He thinks that the above inference should be supported even if

Over-Block($well_2$)

were omitted as a premise about the target, and even if

Sand-Nearby($well_1$)

were omitted as a premise about the source. If the target case contains more indicators of the conclusion and fewer counter-indicators than the source case, thinks Clark, then the conclusion should be preserved. Clark formalized this reasoning, but did not attempt to justify it, deductively or inductively.

1.3 The Reference Class

The account of statistical reasoning I have in mind is due to Kyburg [Kyburg61,74,82], though there may be similar reconstructions with Pollock's competing theory [83,84]. What is crucial about these theories is that they provide a logic for determining what is called "the reference class." The account of analogy I am considering could not be given in theories of enumerative induction, such as considered by Russell. Nor could it be given in a Bayesian theory of statistical inference in which determination of the reference class plays no role.

The problem of determining the reference class begins with explicit acknowledgement of multiple statistical sources. Essentially, choosing the reference class amounts to choosing the right statistical source on which to base inductive judgement. If we think of Neyman-Pearson statistics, the closest kin to the problem of choosing the reference class is the problem of testing for significant difference in two populations. But the statistician's logic is incomplete; most often, a single, homogeneous sampling population is assumed.

When there is information from multiple populations, or multiple classes, choice of the reference class depends on how relevant each class is, and how precise the statistical information. If all statistical information is sufficiently precise, then we try to use information about the most relevant class for which there is statistical information. The Bayesians part company here. Bayesians assume that there is precise statistical information about all classes; choice is a matter of maximum relevance. In Kyburg and Pollock, there may be no statistical information about the class of maximum relevance; of those for about which there is information, there may be multiple classes of maximal relevance.

It may puzzle why we are interested in choosing among multiple statistcial sources when analogy reasons from a

single source. Simply, analogy proceeds with a single source of imperfect match when there may be alternative sources and alternative matches. "Why this analogy?" may be closely related to "Why this class as reference class?"

Kyburg with 1-Projectibility	Clark	Russell
as few as one case suggests mass behavior	one case produces generalization	one case produces generalization
defeasibly	indefeasibly	indefeasibly
multiple classes	multiple determinations	multiple determinations
more specific classes are more relevant	no differential relevance	determinations have ideal relevance
multiple source cases	one source case	one source case

Kyburg	Subjective Bayes	Sampling
multiple cases or known frequencies suggest mass behavior	subjective estimates suggest mass behavior	multiple cases suggest mass behavior
defeasibly	without acceptance	at confidence
multiple classes	one ideally relevant source	no guidance on appropriate population
more specific classes are more relevant	condition on all relevant properties	no concept of relevance
multiple source cases	no source case	multiple source cases

2 Constructions.

2.1 Kyburg's Logical Foundations of Statistical Inference.

Kyburg's account is widely acknowledged among philosophers of science, but is quite complex; I assure the reader that this paper contains no more than the parts of theory required to follow the later discussion.

Kyburg formalizes reasoning about probabilities given statements about relative frequencies among classes.

Let me write

$$[|A|] \text{ for } \{x : A(x)\}.$$

That means $[|A \wedge B|]$ stands for $\{x : A(x) \wedge B(x)\}$. Also, let

$$\%([|A|], [|B|])$$

be the per cent of $[|A|]$'s among $[|B|]$'s.

Consider an example with three statistical sources of varying precision and relevance:

$$\%([|\text{Mets-Game}|], [|\text{Mets-Win}|]) = [.6, .7]$$
$$\%([|\text{Mets-Game} \wedge \text{Night-Game}|], [|\text{Mets-Win}|]) = [.5, .65]$$
$$\%([|(\lambda x).(x = \text{Tuesday's-Game})|], [|\text{Mets-Win}|]) = [0, 1]$$
$$\text{Tuesday's-Game} \in [|\text{Mets-Game}|]$$
$$\text{Tuesday's-Game} \in [|\text{Night-Game}|]$$
$$\underline{\text{Tuesday's-Game} \in [|(\lambda x).(x = \text{Tuesday's-Game})|]}$$
apparently,
$$\text{PROB}(\text{Tuesday's-Game} \in [|\text{Mets-Win}|]) = [.5, .65]$$

The reference class is $[|\text{Mets-Game} \wedge \text{Night-Game}|]$, the most specific class about which adequate statistics are known.

In the example, there are three candidates for the reference class (or candidate reference classes), for the query

$$\text{PROB}(\text{Tuesday's-Game} \in [|\text{Mets-Win}|]) = ?$$

That is, there are three sets that satisfy the syntactic requirements required for projecting a Mets-Win. These sets are

$[|\text{Mets-Game}|]$
$[|\text{Night-Game}|]$
$[|(\lambda x).(x = \text{Tuesday's-Game})|]$.
i.e., $\{x : x = \text{Tuesday's-Game}\}$

The last is the most specific class, but the relevant frequency is not well known among this class. Of the other two frequencies reported, there is conflict: *i.e.* [.6, .7] and [.5, .65] do not stand in the sub-interval relation, and the interval [.5, .65] is associated with the more specific class.

Formally, Kyburg takes an "inference structure" for $\text{PROB}(x \in V)$ to be the collection:

$$< x, A, V, [p, q] >$$

(I am suppressing Quinean quotes, but note that (x) is the universal quantifier in the object language, and the universal quantifier in the meta-language is spelled out, *for all* x; similarly, $\rightarrow$ is the material conditional in the object language, and "*if* ... *then* ..." is the material conditional in the meta-language).

when it is known that

$$x \in A$$
$$\%(A, V) = [p, q]$$
$$A \text{ MAY-PROJECT } V,$$

(Also note that the first two sentences should be properly rendered

$$\ulcorner x \in A \urcorner \in \text{KBASE}$$
$$\ulcorner \%(A, V) = [p, q] \urcorner \in \text{KBASE}$$

to make them metalinguistic assertions, like the third sentence;

⌜A⌝ MAY-PROJECT ⌜V⌝;

I will resist this much propriety).

where the last requirement is that A is a candidate reference class for V, *i.e.* A may project V. I will call V the target property and x the target individual. There are lots of classes that will not be candidate reference classes for V. For instance,

$$V \cup \{x\}$$

is clearly not a candidate reference class in general. This restriction is motivated by the same intuitions that lead to restrictions on what similarities may support analogy. We might not want [|Bob-Hope-did-not-attend|] to be a candidate reference class for [|Mets-Win|]. Similarly, we might not take Russell's P, the property that a and b share in virtue of which an analogy is made, to be the property of not being identical to Bob Hope.

It is an axiom of Kyburg's system that

for all A, B: *if* (A MAY-PROJECT V)
and
(B MAY-PROJECT V)
then (A ∩ B) MAY-PROJECT V .

Given a collection of inference structures, $\{IS_1, IS_2, \ldots\}$, there will be relations of "domination" among them. In the example above,

DOMINATES (
< Tuesday's-Game,
[|Mets-Game ∧ Night-Game|],
[|Mets-Win|], [.5, .65] > ,
< Tuesday's-Game,
[|Mets-Game|],
[|Mets-Win|], [.6, .7] >
).

Let

$$\text{INF-STRUCTS}_{x,V}$$

be the set of inference structures for PROB(x ∈ V); there will be one inference structure per (relevant) statistical source. Also, let

$$\text{REFCLASS}_{x,V}$$

be the reference class for PROB(x ∈ V). When x and V are understood, we drop the indexes.

It is a theorem that

exists p, q *s.t.*
<x, REFCLASS, V, [p, q]> ∈ INF-STRUCTS,
and
not (
exists IS' *s.t.*
IS' ∈ INF-STRUCTS
and
DOMINATES(IS', <x, REFCLASS, V, [p, q]>));

in other words, the inference structure that lists the reference class is undominated in the set of inference structures. Call the inference structure that lists the reference class the "reference inference structure." Say that two inference structures disagree when their intervals don't nest, that is,

DISAGREES (
<x, A, V, [p, q]>,
<x', A', V', [p', q']>
)
iff
¬([p, q] ⊂ [p', q'])
and
¬([p', q'] ⊂ [p, q]) .

Let I-REFCLASS(S) be the reference inference structure for a particular set of inference structures, S, which may not be the same as INF-STRUCTS. Note that I-REFCLASS(INF-STRUCTS) is REFCLASS.

The following two theorems for Kyburg's system establish how much of Clark's monotonicity of conflict resolution we will believe. The first says that if we remove a statistical source that disagrees with our ultimate conclusion, then the ultimate conclusion remains the same. The second says that if we add a statistical source that does not disagree with our ultimate conclusion, then either the conclusion is unchanged, or the new source becomes the new conclusion.

if
(IS₁ ∈ INF-STRUCTS)
and
DISAGREES(IS₁, REFCLASS)
then
not (
DISAGREES(I-REFCLASS(INF-STRUCTS — IS₁), REFCLASS)
).

And

if
¬(IS₂ ∈ INF-STRUCTS)
and
¬DISAGREES(IS₂, I-REFCLASS(INF-STRUCTS))
then
I-REFCLASS((INF-STRUCTS ∪ IS₁))
= I-REFCLASS(INF-STRUCTS)
= REFCLASS
or
I-REFCLASS((INF-STRUCTS ∪ IS₁))
= IS₂ .

Incidentally, this system has been successfully implemented and studied as a system of evidential reasoning [Loui86,88]. These papers also discuss in more detail the combinations that are appropriate in situations in which there is more than one undominated class. Combination of inference structures would take us too far afield, and there is no analogue in analogical reasoning. It suffices to say that here is another facet of statistical reasoning that has not yet been exploited in case-based reasoning (though Clark's attempts are a move in this direction).

2.2 Reconstruction of Russell's Inference.

Armed with this much understanding of Kyburg's system, we can now state Russell's and Clark's analogies as special cases of inferences with this system.

The point is that analogical inferences can be justified and understood in terms of an existing and accepted system of statistical reasoning.

The constructions for Clark's inferences are more interesting, so do not be dismayed by the triviality of claim 1.

> Claim 1. Russell's analogies can be understood as Kyburgian statistical inference given an example from the most specific candidate reference class, and given knowledge that the relative frequency of the target property is extreme in this class.

$P(x, y)$ DETERMINES $Q(x, z)$
$P(a, y) \wedge Q(a, z)$
$\underline{P(b, y)}$
thus,
$Q(b, z)$

becomes

1.1.1. *for all* y, z:
$[|(\lambda x).P(x, y)|]$ MAY-PROJECT $[|(\lambda x).Q(x, z)|]$
and
for all A, t:
if
$(t \in [|(\lambda x).P(x, y)|])$
and $(t \in A)$
and A MAY-PROJECT $[|(\lambda x).Q(x, z)|]$
then
$[|(\lambda x).P(x, y)|] \subseteq A$

1.1.2. $(y)(z)$.
$\%([|(\lambda x).P(x, y)|], [|(\lambda x).Q(x, z)|]) = [1, 1]$
$\vee$
$\%([|(\lambda x).P(x, y)|], [|(\lambda x).Q(x, z)|]) = [0, 0]$

1.2. $a \in [|(\lambda x).P(x, y_1) \wedge (\lambda x).Q(x, z_1)|]$

$\underline{1.3.\ b \in [|(\lambda x).P(x, y_1)|]}$

thus,
1.4. PROB$(b \in [|(\lambda x).Q(x, z_1)|]) = [1, 1]$

(the numbering is suggestive of the mapping between the premises).

Proof of Claim 1. Sufficiency. Without assuming details of Kyburg's system, I can only sketch the proof. The critical observations are that

- 1.2 and 1.1.2 fix the % of Q's among P's.
- 1.3 says that this leads to an inference structure for the probability that b is a Q.
- There may be other disagreeing inference structures, but 1.1. guarantess that they are dominated.

Note that here as in Russell's account, the source case makes the inference about the target indefeasible. We have taken the probabilities to be extremes. If b is known not to be $(\lambda x).Q(x, z_1)$, then that is an inconsistency, not a defeater of the analogical inference.

1.1.1 can be weakened by taking just its first conjunct. 1.4 would still be a conclusion if there are no known competing inference structures for b's Q-ness or $\neg$Q-ness. But note that Russell requires that the predicate P reflect "all the information relevant to the query, ... for example, ... all the factors that might affect the language a person speaks (nationality, country of residence, parents' language, ... and so on) ...." This is reflected in the requirement in (1.1.1) that $[|P(x, y)|]$ be the most specific candidate reference class to which any individual will be known to belong. We require any individual, not just the target individual, because Russell's determination relation says nothing about a particular individual. Usually, $[|(\lambda x).(x = t)|]$ is taken to be a candidate reference class in Kyburg's system, for all queries, but this almost surely will not be the case here. There has to be some reason why, when we do analogy, we are so certain that the inference from the putative determining case will not be defeated by an inference from an even more similar determining case. In Russell, and here, this certainty is simply imposed by fiat. The assumption of certainty is relaxed in Clark's study.

Russell worries about what should happen when Louis speaks two languages, or has two nationalities, and he alters the definition of a determination several times to account for this case. Here, too, there would be complications, but I will ignore them and focus only on total determinations. Accounts of his other determination relations could be given, but would just be distracting.

2.3 Reconstruction of Clark's Inference.

> Claim 2. Clark's analogies can be understood as Kyburgian statistical inference in the presence of multiple statistical sources, when there is knowledge about how conflicts were decided for the source case.

It cannot be understood as using the single case to determine the frequency of the target property among the intersection of candidate reference classes. This is because even if the single case could determine the frequency, and we could project from this single case, we could not explain Clark's insistence on the monotonicity of conflict resolution.

$A_1(x)$ INDICATES $\neg Q(x)$

$A_2(x)$ INDICATES $Q(x)$
$A_3(x)$ INDICATES $Q(x)$
$A_4(x)$ INDICATES $\neg Q(x)$
$Q(a)$
$A_1(a) \wedge A_2(a) \wedge A_3(a) \wedge A_4(a)$
$A_1(b) \wedge A_2(b) \wedge A_3(b) \wedge A_4(b)$
thus,
$Q(b)$

becomes

2.1.1. $\%([|A_1|], [|Q|]) = [0, \epsilon]$
2.1.2. $[|A_1|]$ MAY-PROJECT $[|Q|]$

2.2.1. $\%([|A_2|], [|Q|]) = [1 - \epsilon, 1]$
2.2.2. $[|A_2|]$ MAY-PROJECT $[|Q|]$

2.3.1. $\%([|A_3|], [|Q|]) = [1 - \epsilon, 1]$
2.3.2. $[|A_3|]$ MAY-PROJECT $[|Q|]$

2.4.1. $\%([|A_4|], [|Q|]) = [0, \epsilon]$
2.4.2. $[|A_4|]$ MAY-PROJECT $[|Q|]$

2.5. $a \in [|A_1 \wedge A_2 \wedge A_3 \wedge A_4|]$

2.6. Q(a) is acceptable on 2.1 – 2.5;
i.e. PROB$_{2.1-2.5}(a \in [|Q|]) \geq 1 - \epsilon$

2.7. $b \in [|A_1 \wedge A_2 \wedge A_3 \wedge A_4|]$

thus,
2.8. PROB$(b \in [|Q|]) \geq 1 - \epsilon$

Proof of Claim 2. Sufficiency. The crucial observations are

- There are as many inference structures for $b \in [|Q|]$, as for $a \in [|Q|]$, given 2.1 – 2.5.
- There may be other reasons for $a \in [|Q|]$, but on the force of the inference structures that 2.1 – 2.5 support, PROB$(a \in [|Q|])$ is high.
- So some inference structure with reference class Y, with %(Y, [|Q|]) high, dominates other inference structures (whether this reference inference structure is based on 2.2, 2.3, or some combination of them).
- If this inferece structure is dominating for $a \in [|Q|]$, then it is dominating for $b \in [|Q|]$.

What is more interesting than sufficiency is the insufficiency of a weakened 2.6. If 2.6 is weakened, so that Q(a) is simply acceptable, then this doesn't guarantee 2.8. Q(a) may be acceptable because of some other inference structure for $a \in [|Q|]$, and there may be no analogous inference structure for $b \in [|Q|]$.

Note that monotonicity of conflict resolution is achieved through the theorems on how I-REFCLASS behaves under unions and subtractions from INF-STRUCTS. If inference structures dominate each other in such a way that warrants concluding PROB$(b \in [|Q|])$ exceeds some threshold, then adding yet another inference structure that agrees with that conclusion cannot force the threshold lower. This corresponds to Clark performing the analogy from a source case that has fewer factors indicating the ultimate conclusion than the target case, but just as many counter-indicating factors. Also, if we remove from the set of inference structures an inference structure that disagrees with the conclusion about PROB$(b \in [|Q|])$, that too cannot lower the threshold. This corresponds to Clark performing the analogy from a source case that has as many indicating factors for the ultimate conclusion, but even more counter-indicating factors.

It is tempting, but incorrect to take the translation to be

3.1.1. $\%([|A_1|], [|Q|]) = [0, \epsilon]$
3.1.2. $[|A_1|]$ MAY-PROJECT $[|Q|]$

3.2.1. $\%([|A_2|], [|Q|]) = [1 - \epsilon, 1]$
3.2.2. $[|A_2|]$ MAY-PROJECT $[|Q|]$

3.3.1. $\%([|A_3|], [|Q|]) = [1 - \epsilon, 1]$
3.3.2. $[|A_3|]$ MAY-PROJECT $[|Q|]$

3.4.1. $\%([|A_4|], [|Q|]) = [0, \epsilon]$
3.4.2. $[|A_4|]$ MAY-PROJECT $[|Q|]$

3.5. $\%([|A_1 \wedge A_2 \wedge A_3 \wedge A_4|], [|Q|]) = [1 - \epsilon, 1]$
$\vee$
$\%([|A_1 \wedge A_2 \wedge A_3 \wedge A_4|], [|Q|]) = [0, \epsilon]$

3.6. $a \in [|A_1 \wedge A_2 \wedge A_3 \wedge A_4 \wedge Q|]$

3.7. $b \in [|A_1 \wedge A_2 \wedge A_3 \wedge A_4|]$

thus,
3.8. PROB$(b \in [|Q|]) \geq 1 - \epsilon$.

First, the single case reported in (3.6) does not establish which of the disjuncts in (3.5) is true. Suppose, though, on the basis of (3.6), one of the disjuncts could be made so probable that it is acceptable, through the observation of the target case. This requires an additional theorem about sampling: that most of the candidate reference classes for some property y that are known to be mostly y or mostly $\neg$y, can be decided (with high probability) by looking at a single case. This is a theorem for appropriate ϵ and δ.

3.5.1. (y).
$\%($
$\{ x : (\%(x, y) = [0, \epsilon] \vee \%(x, y) = [1 - \epsilon, 1]) \wedge \text{sample}(x, y) = <1, 1> \},$
$\{ x : \%(x, y) = [1 - \epsilon, 1] \}$
$) \geq 1 - \delta$.

(where, by sample(P, Q) = <s, r>, we mean that s P's were sampled, of which r were Q's and s – r were $\neg$Q's).

Additional knowledge about particular x's and y's, or for particular sampling procedures, might make δ even smaller, for given ϵ.

When we have a situation like (3.5), and the δ for which (3.5.1) is a theorem is small enough to suit our needs (*i.e.* 1 − δ is above the threshold of acceptability), then let us write that Q is single-case-projectible from $A_1 \wedge \ldots \wedge A_4$; that is:

$$[|A_1 \wedge \ldots \wedge A_4|]_{1\text{-PROJECTS}}\,[|Q|].$$

Then (3.6) decides which disjunct in (3.5) is the right one; the mass of $A_1 \wedge \ldots \wedge A_4$ is either Q or $\neg$Q, and since a is a Q, this decides that the bulk of $A_1 \wedge \ldots \wedge A_4$ is Q with high probability. So Q(b) could be inferred.

But the monotonicity of conflict resolution would be violated for the following reason. If

$$\%(\,[|A_1 \wedge A_2 \wedge A_3 \wedge A_4|],\,[|Q|]\,) = [1-\epsilon,\,1]$$
$$\wedge$$
$$\%(\,[|A_1|],\,[|Q|]\,) = [0,\,\epsilon],$$

it does not follow that

$$\%(\,[|A_2 \wedge A_3 \wedge A_4|],\,[|Q|]\,)$$

can be bounded. In particular, it is not necessarily close to one. If $b \in [|A_1 \wedge A_2 \wedge A_3 \wedge A_4|]$, indeed, Q(b) can be projected. But if (3.7) is altered, if b is only known to be in $[|A_2 \wedge A_3 \wedge A_4|]$, no projection is possible. It could be that most races in which there are Porsches are not close races, most races in which there are Ferraris are not close races, but races in which there are both Porsches and Ferraris are often close races.

So in this formulation of the inference, we cannot weaken the number of counter-indicating factors for the source case. It may be that the interaction of two counter-indicating factors produces a joint indicating factor. Why could we weaken it in the first reduction of Clark? In that reduction, the combination of two counter-indicating inference structures can only counter-indicate. So the counter-indicators for the source could be weakened without disturbing the conclusion. Combination of inference structures is related to the cross product of two sets, while joint effects are related to the intersection of two sets; therein lies the difference.

Anyway, we do allow the other half of Clark's monotonicity desideratum. We can know one more indicating factor about b, the target, and continue to draw the conclusion. Add

3.7.1. $b \in [|A_1 \wedge A_2 \wedge A_3 \wedge A_4 \wedge A_5|]$
3.7.2. $\%(\,[|A_5|],\,[|Q|]\,) = [1-\epsilon,\,1]$
3.7.3. $[|A_5|]$ MAY-PROJECT $[|Q|]$

and (3.8) is still a conclusion in Kyburg's system.

3 Discussion.

3.1 Considerations on Clark's Defeasible Analogical Reasoning.

It is fallacious to propose a reduction of one inference system as a special case of another, then criticize it because it appears to make assumptions in the language of the reducing system. It is fallacious because there is no guarantee that the reduction is the only possible, and there is no guarantee that ensuing disputes are not defects of the reducing system. This is what the Bayesians are sometimes guilty of doing when they assail Dempsterian inference.

I will not do this. But there are some claims about Clark's analogical inferences I would like to make, and I take the reductions to be merely suggestive of the truth of those claims.

The reason I have belabored the second formulation of Clark-like inference is that it is an epistemologically more satisfying account of analogy. It does not capture both halves of Clark's monotonicity of conflict resolution. But it is simpler to use the second reduction and not insist on the ability to project from a source to a target when more relevant things are known about the source than about the target (namely, that the source has additional counter-indicating factors of the ultimate conclusion).

The latter account requires trivial background knowledge; it requires almost no assumptions in order to reconstruct Clark's inferences. Domination of inference structures based on specificity of candidate reference classes does all the work. There are only two specializations. The first is that when two factors both indicate a conclusion, they do so with frequency intervals that nest. That is, if A_1 indicates Q, and A_2 indicates Q, then $\%(A_1, Q)$ and $\%(A_2, Q)$ should stand in the (possibly improper) subinterval relation. I guaranteed this above by taking all the intervals for indicating factors to be $[1-\epsilon, 1]$ (and all the intervals for counter-indicating factors to be $[0, \epsilon]$). This could have been done in a more general way: by taking all the intervals for indicators to be anchored on the right at 1 (and taking all the intervals for counter-indicators to be anchored on the left at 0). The second specialization is in (3.5.1), which says that observation of a single case determines the statistics to be on one extreme or the other, hence allows projection from that single case. (3.5.1) is a very natural assumption about sampling, for the appropriate δ.

In the former account, which preserves both parts of the monotonicity of conflict resolution, the source case had to be treated in a special way. It was not enough to say in (3.6) that Q(a) was in fact accepted, for instance, by some fortuitous observation. It had to be that Q(a) was acceptable, on the basis of a's having properties A_1, A_2, A_3, and A_4. Q(a) could then be accepted by inference. Or perhaps that inference was unnecessary because Q(a) had already been observed; it wouldn't matter which. But it is not enough to say that Sand-Nearby, and Unfavorable-Environment, and Sand, among others, all co-occurred at $well_1$. It must be that their co-occurrence was not accidental, that this was a special kind of co-occurrence that determines how conflicting arguments of exactly the kind that were involved are to be resolved, for all future cases, including $well_2$. It must be that the co-occurrence of Louis's

Nationality and his Language is not spurious; rather, that Louis's Nationality and Louis's Language's co-occur representatively.

3.2 Defeasible Determination and Defeasible Representativeness.

In the case of Russell-like analogies, since there is no defeasibility, the determination relation dictates that any source example will relate Nationality and Language in a representative way.

In Clark-like analogies, the stipulation of this representativeness is the intuition that drives the design of the inference system. Any source example that involves certain factors and their resolution is representative of all such resolutions.

Clark puzzles over how to generalize the formalism so that cases with the same indicators and counter-indicators could have resolved differently in past, arriving at different ultimate conclusions. He has taken the first step of introducing defeasibility into determinations. Now a factor, such as French-Nationality, only appears to determine whether French-Language or ¬French-Language; it is a prima facie determiner. It defeasibly determines, and the ultimate conclusion about whether French-Language or ¬French-Language in a particular case depends on what other defeasible determinations play mitigating roles, such as ¬French-Nationality-Parents defeasibly determining whether French-Language or ¬French-Language.

Clark puzzles over the next step, introducing defeasibility into representativeness. How should conflict resolution be done defeasibly?

We may consider the inference

$A_1(x)$ INDICATES $\neg Q(x)$
$A_2(x)$ INDICATES $Q(x)$
$A_3(x)$ INDICATES $Q(x)$
$A_4(x)$ INDICATES $\neg Q(x)$
$A_1(a_1) \wedge A_2(a_1) \wedge A_3(a_1) \wedge A_4(a_1) \wedge Q(a_1)$
$A_1(a_2) \wedge A_2(a_2) \wedge A_3(a_2) \wedge A_4(a_2) \wedge Q(a_2)$
$A_1(a_3) \wedge A_2(a_3) \wedge A_3(a_3) \wedge A_4(a_3) \wedge \neg Q(a_3)$
$\underline{A_1(b) \wedge A_2(b) \wedge A_3(b) \wedge A_4(b)}$
thus,
$Q(b)$?

Clark mentions that a "majority verdict" could be taken, in this case, two cases, a_1 and a_2, are in favor of Q, while the single case, a_3, is opposed.

Are we willing to choose Q over ¬Q for b when no overwhelming number of past cases are examples of Q? I think we are not. If the number of cases for Q were overwhelming, say 10 against 1, then perhaps we should project Q(b). Otherwise, by my lights, there is no sense to the inference, even if possibly construed as some strong form of analogy.

The only sense I can make of such an inference is that we are accumulating instances of the class $[|A_1 \wedge A_2 \wedge A_3 \wedge A_4|]$, and observing the relative frequency among them that is is $[|Q|]$. Our willingness to project from this class based on a small number of cases includes those times when we have a sample of 1 among 1, or 2 among 2, perhaps even 9 among 10; but not 2 among 3, or 5 among 11.

Fully defeasible analogical reasoning is apparently just defeasible reasoning about the reference class, with the right to project from certain small samples.

3.3 Defeasible Reasoning with Specificity.

Oddly, Clark makes no use of the idea of specificity, which pervades defeasible statistical reasoning, and defeasible reasoning in general [Loui87]. If A_1 indicates Q, while $A_1 \wedge A_2$ counter-indicates Q, the latter should supercede the former for a case t, *s.t.* $A_1(t) \wedge A_2(t)$. This is not a particularly egregious oversight, since Clark's whole philosophy is to make no a priori distinctions among determining factors; he wants to resolve all conflicts by looking for source cases that manifest these factors and seeing how their conflicts were resolved. But he misses an opportunity by not seeing the source case as a more specific reason than the indication relation.

He could be faced with the case above, where A_2 and A_3 are both indicators of Q, and A_1 and A_4 are both counter-indicators of Q. There are two potential sources: a_1 and a_2. a_1 is an A_1 and an A_2, as well as a $\neg Q$. a_2 is an A_1 and an A_2 and an A_4, as well as a Q. In this case, though one source argues for Q and the other for ¬Q, so that there is no majority verdict, taking specificity seriously would demand the (defeasible) conclusion Q(b). a_2 is just more specific a source than a_1.

3.4 Case-Based Generalization versus Case-Based Projection.

The difference between generalization and projection for arbitrary cases is that the projection is defeasible. It is one thing to infer

P DETERMINES Q
$\underline{P(a) \wedge Q(a)}$
thus,
$(x).\ P(x) \rightarrow Q(x)$

and quite another to say

P DETERMINES Q
$\underline{P(a) \wedge Q(a)}$
thus,
for all x: *if* P(x) is known,
then Q(x) is a justified defeasible conclusion .

The two are not the same because when $\vdash$ is taken to be non-monotonic, the deduction theorem disappears. The former conclusion indefeasibly yields Q from P. The latter admits defeaters.

What really seems to be desired is the latter.

"Case-based generalization" is a misnomer. Clark and Russell understand the logic of analogy to be monotonic. I think this is a mistake.

4 Control of Statistical Reasoning Driven by Few Cases.

The possibility of case-based reasoning suggests new control strategies for statistical reasoning programs. Already there is a need for studying control issues in programs that pore over their database in response to a query, to construct samples, from which to do reasoning about reference classes (*e.g.* the program described in [Loui88]). It makes little sense to continue to extend sample sizes beyond 10, or 20, in commonsense reasoning. There is hardly a difference in the intervals that result from a sample of 7 out of 10, and a sample of 14 out of 20. If I had time to examine 20 cases, I would spot at 7 out of 10 for one class, and spend the rest of my counting time in its subclasses.

A meta-level utility analysis of the expected value of continued examination of cases is required, no doubt (see for example, [Russell&Wefald88]).

Prior to doing such an analysis, we should identify what choices make sense. These choices are determined by what are our current best arguments for projecting some property, and what counter-arguments, if constructible, would be effective rebuttals. Our theory of analogy, or of more general projection from few cases, is what determines where to seek counter-arguments.

Consider projecting from the single case:

Porsche(944) ∧ Powerful(944)
Porsche(924)
apparently,
Powerful(924)

There are a few ways in which search can now be directed. A counter-argument to this projection could produce counter-examples to the co-occurrence, *e.g.*

Porsche(356) ∧ ¬Powerful(356)

which dilutes the relevant statement of relative frequency. Or it could produce a property that distinguishes putative target and source, *e.g.*

VW-Project(924) ∧ ¬VW-Project(944)

together with a statement that this property is a counter-indicating factor, that is, that it indeed leads to an interfering inference structure

VW-Project(Bug) ∧ ¬Powerful(Bug) .

This amounts to an appeal to the target's membership in a more specific (*i.e.* more relevant) class that is apparently counter-indicating.

A third way to attack the argument is to attack the determination relation in virtue of which the single case can be projected. For instance, if one could produce

Ford(Mustang) ∧ Powerful(Mustang)
Ford(Escort) ∧ ¬Powerful(Escort)

it would be contentious whether

for all x, y:
if x ∈ Car-Manufacturers
and y ∈ Performance-Features
then x 1-PROJECTS y .

This amounts to attacking the inference that led to the 1-projectibility relation between Porsche and Powerful, that is being assumed (in the real world, we probably already know this relation, so the Ford examples are not going to sway our opinion).

The Ford example is prima facie evidence that there is a Car-Manufacturer class that is inhomogeneous in some Performance-Feature class. This example could be countered by identifying Porsche in a subclass, *e.g.* European-Marques, where the Ford example is excluded from this subclass. And there should be prima facie evidence that there is homogeneity of members of this class with respect to kinds of performance features; there should be reason to believe that a Performance-Feature could be 1-projected from a European-Car-Marque:

%([|Saab|], [|Handles|]) = [1 − ϵ, 1]
%([|Yugo|], [|Handles|]) = [0, ϵ]
%([|Lotus|], [|Brakes-Well|]) = [1 − ϵ, 1]
%([|Daimler-Benz|], [|Slow|]) = [0, ϵ]

This is a lot of background knowledge, and in the presence of such knowledge we might as well assume that we also know

%([|Porsche|], [|Powerful|]) .

But it should be clear how the logic could direct the dialectic. If any of these counter-arguments could be produced, search could next be directed toward finding a reinstating argument, and so forth.

Investigations of control of this kind of reasoning would also benefit control of purely qualitative defeasible inference, such as [Loui87] and [Pollock87].

How precisely to make control choices appropriate to this kind of reasoning is the subject of a more ambitious invesitigation. For our purposes, it is enough to recognize this kind of reasoning, and to recognize the relation between control of choices and the study of how to undermine analogical inferences.

5 Conclusion.

Modern philosophers of science such as Quine and Ullian [Quine&Ullian70] and Kyburg [Kyburg61] hold that analogy is an uninteresting special case of induction. AI authors have balked at this, preferring to view analogy as

a more interesting species of deductive reasoning, in Russell's case, or as the resolution of conflicting defeasible arguments, in Clark's case. I present middle ground. Analogy's problems are best sorted with our most expressive language and machinery for inductive statistical reasoning. Case-based reasoning reveals itself as the guide to dialectical maneuvers in this statistical reasoning.

6 Bibliography.

Clark, P. "Representing arguments as background knowledge for the justification of case-based inferences," AAAI Workshop on Case-Based Reasoning, St. Paul, 1988.

Davies, T. and Russell, S. "A logical approach to reasoning by analogy," IJCAI, 1987.

Gentner, D. "Structure Mapping," *Cognitive Science 7*, 1983.

Kedar-Cabelli, S. "Purpose-Directed Analogy," Cognitive Science Society Meeting, 1985.

Kyburg, H. *Probability and the Logic of Rational Belief*, Wesleyan, 1961.

Kyburg, H. *Logical Foundations of Statistical Inference*, Reidel, 1974.

Kyburg, H. "The Reference Class," *Philosophy of Science 50*, 1982.

Loui, R. "Computing reference classes," AAAI Workshop on Uncertainty, Philadelphia, 1986.

Loui, R. "Defeat among arguments", *Computational Intelligence 3*, 1987.

Loui, R. "Evidential reasoning in a network usage prediction testbed," AAAI Workshop on Uncertainty, St. Paul, 1988.

Pollock, J. "A theory of direct inference," *Theory and Decision 16*, 1983.

Pollock, J. "Foundations for direct inference," *Theory and Decision 17*, 1984.

Pollock, J. "Defeasible reasoning," *Cognitive Science 12*, 1987.

Quine, W.V.O. and Ullian, J. *The Web of Belief*, Random House, 1970.

Russell, S. Analogical and Inductive Reasoning. Ph.D. dissertation, Stanford University, Dept. of Computer Science. Also STAN-CS-87-1150, 1987.

Russell, S. and Wefald, E. "A Meta-Greedy Game-Playing Algorithm," draft, Computer Science Division, UC Berkeley, 1988.

Weitzenfeld, J. "Valid Reasoning by Analogy," *Philosophy of Science 51*, 1984.

Winston, P. "Learning and Reasoning by Analogy," *CACM 23*, 1980.

Plausible World Assumption

Eliezer L. Lozinskii
Institute of Mathematics and Computer Science
The Hebrew University, Jerusalem 91904, Israel

Abstract

A way of producing beliefs in non-monotonic systems, called *Plausible World Assumption (PWA)*, is presented. PWA has certain advantages compared to Negation as Failure, CWA, GCWA. The PWA provides beliefs for *all atoms* of the Herbrand base of a given system leaving no one of them undefined. Most of the assumptions agree with the *majority of the system models*, and so have a good chance to be approved by the reality. We study a notion of *relative monotonicity*, and show that under PWA a system possesses a high degree of this desirable feature in the sense that most of its beliefs are not affected by any change of other ones.

1 Introduction

In recent years logic became a powerful tool of rapid development of a wide variety of information processing systems, such as Expert Systems, Knowledge-Base Systems, Deductive Databases.

Example 1.1. Consider a set of clauses

$$S = \{ P(a), Q(b), P(x) \rightarrow R(x) \}.$$

S contains two *facts* (that $P(a)$ and $Q(b)$ are true in S), and allows us to derive (by using the *axiom* $P(x) \rightarrow R(x)$) a new fact $R(a)$. On the other hand, S provides no grounds for any decision regarding the truth values of $P(b), Q(a), R(b)$. Actually these values are *undefined* in S, or we could say that they are *unknown* due to the given state of S. □

Well, we may not be satisfied with such uncertainty even though it reflects our restricted knowledge of the real world described by the system. Indeed, we may need some knowledge or estimate of $P(b)$ for our further activity. So, we may try to guess or make an assumption about the value of $P(b)$, which corresponds to our beliefs provided that the beliefs are consistent with S. For instance, assuming that $P(b)$ is true implies that $R(b)$ is true either, or we may believe that both $P(b)$ and $R(b)$ are false. Thus, we would like to have a systematic approach to forming our beliefs about unknown facts.

The widely adopted approaches, such as *Negation as Failure* [Clark, 1978], *Closed World Assumption, CWA* [Reiter, 1978], *Circumscription* [McCarthy, 1980], reflect the idea of minimizing sets of positive facts in the system assuming *negation by default* (cf. [Reiter, 1980, Gabbay, 1986]), which can be phrased as "If ϕ cannot be proved true in S, then ϕ should be assumed false in S", or "The only facts that are true in S are the ones that can be derived from S", or "If assuming that ϕ is false is consistent with S, then ϕ is believed false in S". So, in Example 1.1 each one of $P(b), Q(a), R(b)$ should be assumed false.

Very often negation by default is in accord with real cases. For instance, if a person name John Smith does not appear in a company employees list, and the fact of his employment cannot be derived from the available data, then John Smith is indeed not employed by the company. But in general, assuming that all we don't know about in the world (which is quite a lot) is false, must lead to conflict with the reality. If a system represents a large and complex (and probably changing) domain, then the knowledge of the system designers regarding the domain may not be exhaustive, making negation by default not applicable. Negation by default may be viewed as a claim of our omniscience. However, as we learn more about real world, it may turn out that some of our beliefs are wrong, and this discovery in its turn may disprove conclusions that have been drawn from the wrong beliefs.

This research is supported in part by Israel National Council for Research and Development, grants 2454-3-87, 2545-2-87.

Example 1.2. Consider again Example 1.1 where $P(b), Q(a), R(b)$ cannot be proved in S:

$$S \nvdash P(b), \quad S \nvdash Q(a), \quad S \nvdash R(b).$$

(i) Suppose that according to CWA we assumed that both $P(b)$ and $R(b)$ are false in S, but later on have discovered that $P(b)$ is actually true. Then we have to change *both* our beliefs about $P(b)$ and $R(b)$. Any consequences of these beliefs must also be checked and probably corrected.

(ii) We may have assumed that both $P(b)$ and $R(b)$ are true, which is consistent with S. If it turns out afterwards that $R(b)$ is false, then again *both* our beliefs regarding $P(b), R(b)$ must be changed.

(iii) Now suppose we believe that $P(b)$ is false, but $R(b)$ is true, which is still consistent with S. In this case a discovery that any one of these beliefs is mistaken affects only this particular belief, but no other one in S.

(iv) To exhaust all possible assumptions for $P(b)$, $R(b)$, note that assuming $P(b) = 'true'$ and $R(b) = 'false'$ is inconsistent with S.

Regarding $Q(a)$, any erroneous belief about it affects no other one in S. □

Example 1.2 illustrates an important feature of systems with beliefs, which is their *non-monotonicity* [Bossu and Siegel, 1985, Gabbay, 1985, McCarthy, 1980, Minker, 1982, Reiter, 1980]. If a system, S, adopts beliefs about formulae not provable in S, then new information, C, being consistent with S may contradict some previous beliefs and their consequences. So, it may happen that, for instance, D is true in S but false in $S \cup C$.

Example 1.2 shows that possible assumptions in a given system differ in their stability against discovery of wrong beliefs, or in other words, in the degree of their non-monotonicity. In this sense we would prefer assumption (iii) to (i) or (ii).

2 Positive and negative facts

Consider a system, S, as a set of clauses. If a ground atom $P(\bar{a})$ ($\bar{a}$ is a vector of constants) can be derived from S, $S \vdash P(\bar{a})$, then $P(\bar{a})$ is a *positive fact* in S; if $S \vdash \neg P(\bar{a})$, then $P(\bar{a})$ is a *negative fact* in S. Non-ground clauses are usually referred to as *axioms*, and often presented in the form of implication. We don't presume that the form of an axiom imposes any priorities on its literals (cf. [Bidoit and Hull, 1986, Przymusinski, 1986]).

A (*Herbrand*) *interpretation*, I, for a system S is a set of literals, $I = \{L\}$, such that for every ground atom A of the Herbrand base of S, I contains either A (which means that A is true in I) or $\neg A$ (meaning that A is false). A *model* μ of S is an interpretation satisfying S. Let POS, NEG denote, respectively, sets of positive and negative literals of μ. A model $\mu_1 = POS_1 \cup NEG_1$ is a *minimal model* of S (cf. [Minker, 1982]) if there exists no model $\mu_2 = POS_2 \cup NEG_2$ such that $POS_2 \subset POS_1$. For instance, $P(a), P(b), Q(a), Q(b), R(a), R(b)$ is a model of the system of Example 1.1, while $P(a), Q(b), R(a)$ is its minimal model, where the negative literals are not shown explicitly. In the sequel minimal models will be identified with the sets of their positive literals.

So, a minimal model of a system S determines a minimum set of *positive* facts necessary for satisfying S. Such a preference for positive facts reflects the situation that relations are usually defined in a system in such a way that there is fewer positive facts than negative ones, and so just the positive facts are explicitly stored, thus minimizing the required storage space. For example, consider a *complete* list of faculty presented by relation $FAC(name, department)$ that stores *positive* facts regarding the faculty. If a person name does not appear in FAC, it means, in the spirit of CWA, that the person is not a member of the faculty. So, there is no need to store a large quantity of *negative* facts stating that, say, all the students (who usually outnumber the faculty) are not faculty members.

But suppose that only a *partial* list of the faculty is available, $PARTFAC(name, department)$. If trying to find out the affiliation of Dr. Bright (not appearing in $PARTFAC$) we make a telephone inquiry, we may collect a number of negative facts, like $\neg PARTFAC(Bright, Physics)$, $\neg PARTFAC(Bright, CS)$, $\neg PARTFAC(Bright, EE)$, etc. As only a partial knowledge of the faculty is available, if an individual does not appear in $PARTFAC$, we cannot make any definite conclusion regarding his affiliation. In such situations representation of certain negative facts becomes as important and meaningful as that of positive ones.

Reiter [1978] has proved that a consistent Horn system is consistent with CWA, but a direct application of CWA to a non-Horn system may lead to inconsistency. To overcome the problem with non-Horn systems Minker [1982] has introduced the *Generalized Closed World Assumption (GCWA)*. Subsequently it has been extended in a number of works [Gelfond *et al.*, 1986,

Yahya and Henschen, 1985].

Definition 2.1. (Semantic definition of GCWA [Minker, 1982]). A ground atom is true in a system S iff it belongs to all minimal models of S. All such atoms constitute the set of *positive definite atoms (PD)* of S. A ground atom can be assumed false in S iff it appears in no minimal model of S. All such atoms constitute the set of *negative definite atoms (ND)* of S. All ground atoms not in $PD \cup ND$ belong to the set of *indefinite atoms (ID)* of S, such that $ID = HB - (PD \cup ND)$, where HB denotes the Herbrand base of S. Indefinite atoms are not assigned truth values under the *GCWA*. □

GCWA classifies atomic formulae by dividing the Herbrand base of a system into three disjoint sets: *PD*, *ND*, *ID*. Following observations are of use in trying to achieve a more detailed classification.

Observation 2.1. As it is known, $S \vdash L$ iff $S \cup \{\neg L\}$ is inconsistent. However, if for an atom A neither A nor $\neg A$ can be proved in S, then there is certain freedom to assume A true or false, since each one of these assumptions is consistent with S. We may try to find out which of the assumptions is better from a certain point of view. □

Observation 2.2. Let *MOD* denote the set of all models of a system S. If S describes adequately a certain part of the real world, then any information about the world must satisfy S, or in other words, conform to one of the models belonging to *MOD*. Consider an atom A that can be assumed either true or false. Let $MOD(A)$, $MOD(\neg A)$ denote sets of models of S in which A is true or false, respectively, such that $MOD = MOD(A) \cup MOD(\neg A)$. If we believe, say, that A is false, but it turns out that A is true in the reality, then we have to correct our belief together with its consequences. Willing to ensure that such corrections don't occur often, and lacking precise knowledge of the world's behaviour we may reasonably assume that the more models of S contain A the more likely A is true in the reality at any moment. So, we should assume $A = 'true'$ if A is true in the majority of the models of S, but $A = 'false'$ otherwise. This approach holds also regarding those atoms whose truth values can be definitely proved in the system, since for these atoms one of the subsets of *MOD* is empty. □

Example 2.1. $S = \{P(a) \vee \neg P(b), P(a) \vee \neg P(c)\}$. This is a Horn system with an empty minimal model such that $PD = \emptyset$, $ID = \emptyset$, $ND = \{P(a), P(b), P(c)\}$, so $P(a), P(b), P(c)$ all are assumed false under *GCWA*.

The system has five models:

$$\mu_1 = \{\neg P(a), \neg P(b), \neg P(c)\},$$
$$\mu_2 = \{P(a), \neg P(b), \neg P(c)\},$$
$$\mu_3 = \{P(a), \neg P(b), P(c)\},$$
$$\mu_4 = \{P(a), P(b), \neg P(c)\},$$
$$\mu_5 = \{P(a), P(b), P(c)\},$$

such that $|MOD(P(a))| = 4$, $|MOD(\neg P(a))| = 1$,
$|MOD(P(b))| = 2$, $|MOD(\neg P(b))| = 3$,
$|MOD(P(c))| = 2$, $|MOD(\neg P(c))| = 3$.

By the argument of Observation 2.2, it is expedient to assume that $P(a)$ is true. And this belief is justified by the set of models even stronger than the assumption that $P(b)$ and $P(c)$ are false. □

3 Kernels vs models

Several models of a system S can be represented by their common part in the following way.

Definition 3.1. Let κ be a partial interpretation of S. Then κ is a *kernel* of S if (a) every interpretation I containing κ, $\kappa \subseteq I$, is a model of S, and (b) no proper subset of κ satisfies (a). We say that a kernel κ *represents* a set $MOD(\kappa)$ of all models of S containing κ. $KER(S)$ denotes a set of all kernels of S such that every model of S is represented by a kernel $\kappa \in KER(S)$. □

Example 3.1. Consider the system of Example 2.1. It has two kernels, $\kappa_1 = \{P(a)\}$, $\kappa_2 = \{\neg P(b), \neg P(c)\}$, representing following models:
$MOD(\kappa_1) = \{\mu_2, \mu_3, \mu_4, \mu_5\}$, $MOD(\kappa_2) = \{\mu_1, \mu_2\}$. □

Theorem 3.1. Let $POS(\kappa)$, $NEG(\kappa)$ denote, respectively, sets of all positive and negative literals of a kernel κ of S. Then

(a) for every minimal model μ there is a kernel κ such that $POS(\kappa) = \mu$;

(b) for every kernel κ' there is a minimal model μ' such that $\mu' \subseteq POS(\kappa')$;

(c) there is no minimal model μ'' and kernel κ'' such that $POS(\kappa'') \subset \mu''$ ($\subset$ denotes a *proper* subset).

Proof. (a) If μ is a minimal model of S then $M = \mu \cup neg(HB - \mu)$ is a model of S, where $neg(SET) = \{\neg L \mid L \in SET\}$. Let M be represented by a kernel $\kappa \in KER(S)$, and so, $\kappa \subseteq M$ and $POS(\kappa) \subseteq \mu$. If $POS(\kappa) \subset \mu$ then μ is not minimal since $POS(\kappa) \cup neg(HB - POS(\kappa))$ is a model of S. Therefore $POS(\kappa) = \mu$.

(b) Since $POS(\kappa') \cup neg(HB - POS(\kappa'))$ is a model of S, there must exist a minimal model μ' of S such that $\mu' \subseteq POS(\kappa')$.

(c) By the same argument, if $POS(\kappa'') \subset \mu''$ then μ is not minimal, contradicting the premise. □

Now, given a system S, we intend to classify the atoms of its Herbrand base, HB, with regard to their occurrence in kernels of S.

Definition 3.2. For all atoms $A \in HB$:

(i) An atom A is a *positive* (or *negative*) *fact* iff literal A (resp., $\neg A$) occurs in all kernels of S. Denote by $POSFCT$, $NEGFCT$, respectively, the sets of all positive and negative facts of S.

(ii) A is a *positive* (or *negative*) *belief* iff literal A (resp., $\neg A$) occurs in some but not all kernels, and $\neg A$ (resp., A) occurs in no kernel. Denote by $POSBEL$, $NEGBEL$, respectively, the sets of all positive and negative beliefs of S.

(iii) A is a *neutral belief* iff neither of literals A, $\neg A$ occurs in any kernel. Denote by $NTRBEL$ the set of all neutral beliefs of S.

(iv) A is a *group belief* iff each literal A, $\neg A$ occurs in some kernel. Denote by $GRPBEL$ the set of all group beliefs of S. □

Example 3.2.

$S = \{ P(a) \vee Q(b), P(x) \vee \neg Q(y) \vee (x=y) \}$.

Although this is a non-Horn system, it has a unique minimal model $\mu_0 = \{P(a)\}$. So, due to $GCWA$, $PD = \{P(a)\}$, $ND = \{P(b), Q(a), Q(b)\}$, $ID = \varnothing$.

There are six models:

$\mu_1 = \{P(a), \neg P(b), \neg Q(a), \neg Q(b)\}$,
$\mu_2 = \{P(a), \neg P(b), \neg Q(a), Q(b)\}$,
$\mu_3 = \{P(a), P(b), \neg Q(a), \neg Q(b)\}$,
$\mu_4 = \{P(a), P(b), \neg Q(a), Q(b)\}$,
$\mu_5 = \{P(a), P(b), Q(a), \neg Q(b)\}$,
$\mu_6 = \{P(a), P(b), Q(a), Q(b)\}$,

represented by two kernels:

$\kappa_1 = \{P(a), P(b)\}$, $\kappa_2 = \{P(a), \neg Q(a)\}$,

such that $POSFCT = \{P(a)\}$, $NEGFCT = \varnothing$,
$POSBEL = \{P(b)\}$, $NEGBEL = \{Q(a)\}$,
$NTRBEL = \{Q(b)\}$, $GRPBEL = \varnothing$. □

Definition 3.3. Let η be a set of literals containing $L \in \{A, \neg A\}$. Then $\bar{\eta}_A$ denotes a set *conjugate to* η *w.r.t.* A, such that $\bar{\eta}_A = (\eta - \{L\}) \cup \{\neg L\}$. □

Lemma 3.1. For all atoms $A \in HB$:

(i) $A \in POSFCT$ (or $A \in NEGFCT$) iff A is true (resp., false) in all models of S.

(ii) If $A \in POSBEL$ then the set of all models of S in which A is true, $MOD(A)$, dominates the set of all models of S in which A is false, $MOD(\neg A)$. If $A \in NEGBEL$, then $MOD(\neg A)$ dominates $MOD(A)$. In this sense, if $A \in POSBEL$ (or $A \in NEGBEL$) then literal A (resp., $\neg A$) occurs in the majority of models of S.

(iii) $A \in NTRBEL$ iff there exists a one-to-one mapping from $MOD(A)$ *onto* $MOD(\neg A)$. So, if the set of all models of S, $MOD(S)$, is finite, then $A \in NTRBEL$ iff A is true in exactly one half of the models of S.

Proof. (i) If $A \in POSFCT$ then by Definition 3.2 literal A occurs in all kernels of S. As every model is represented by a kernel, A is true in all models of S. If $A \notin POSFCT$ then there is a kernel κ such that $A \notin \kappa$, and a model μ represented by κ such that $\neg A \in \mu$. By the same argument, $A \in NEGFCT$ iff A is false in all models of S.

(ii) Let KER_1 stand for the set of all kernels of S containing literal A, and KER_2 denote the set of all kernels of S containing neither A nor $\neg A$. If $A \in POSBEL$, then by Definition 3.2, $KER_2 = KER(S) - KER_1$, $KER_1 \neq \varnothing$, $KER_2 \neq \varnothing$. Denote by MOD_1, MOD_2 sets of models represented by all kernels of KER_1, KER_2, respectively. Then $MOD(S) = MOD_1 \cup MOD_2$. Let $MOD_2(A)$, $MOD_2(\neg A)$ denote the subsets of MOD_2 containing, respectively, literal A or $\neg A$. Since literal A occurs in all kernels of KER_1, we have

$$MOD(A) = MOD_1 \cup MOD_2(A),$$
$$MOD(\neg A) = MOD_2(\neg A).$$

As neither A nor $\neg A$ occurs in any kernel of KER_2, for every model $\mu \in MOD_2$ there is a model $\eta \in MOD_2$ such that $\eta = \bar{\mu}_A$. This establishes a one-to-one mapping from $MOD_2(\neg A)$, and so from $MOD(\neg A)$ *onto* $MOD_2(A)$. The latter is a proper subset of $MOD(A)$, hence $MOD(A)$ dominates $MOD(\neg A)$. If $MOD(S)$ is finite, then $|MOD(A)| > |MOD(\neg A)|$. By the same argument, if $A \in NEGBEL$, then $MOD(\neg A)$ dominates $MOD(A)$, and for a finite $MOD(S)$, $|MOD(\neg A)| > |MOD(A)|$.

(iii) $\rightarrow$: If $A \in NTRBEL$ then $KER_1 = \varnothing$, $MOD_1 = \varnothing$, $MOD(A) = MOD_2(A)$, $MOD(\neg A) = MOD_2(\neg A)$, and hence there is a one-to-one mapping from $MOD(A)$ *onto* $MOD(\neg A)$.

$\leftarrow$: If there is a one-to-one mapping from $MOD(A)$ *onto* $MOD(\neg A)$, then for every model μ of S, its conjugate $\bar{\mu}_A$ is also a model of S. If we suppose now that there is a kernel κ containing literal A or $\neg A$, then there must be a model η represented by κ such that $\bar{\eta}_A$ is not a model of S — a contradiction.

Whence $A \in NTRBEL$.

Thus, if $MOD(S)$ is finite, then $A \in NTRBEL$ iff $|MOD(A)| = |MOD(\neg A)| = \frac{1}{2}|MOD(S)|$. □

Example 3.3. $S=\{P(a) \vee Q(b), \neg P(a) \vee \neg Q(b)\}$. There are 8 models:

$\mu_1 = \{P(a), P(b), Q(a), \neg Q(b)\}$,
$\mu_2 = \{P(a), P(b), \neg Q(a), \neg Q(b)\}$,
$\mu_3 = \{P(a), \neg P(b), Q(a), \neg Q(b)\}$,
$\mu_4 = \{P(a), \neg P(b), \neg Q(a), \neg Q(b)\}$,
$\mu_5 = \{\neg P(a), P(b), Q(a), Q(b)\}$,
$\mu_6 = \{\neg P(a), P(b), \neg Q(a), Q(b)\}$,
$\mu_7 = \{\neg P(a), \neg P(b), Q(a), Q(b)\}$,
$\mu_8 = \{\neg P(a), \neg P(b), \neg Q(a), Q(b)\}$,

represented by two kernels:

$\kappa_1 = \{P(a), \neg Q(b)\}$, $\kappa_2 = \{\neg P(a), Q(b)\}$,

which implies the following classification:

$POSFCT = NEGFCT = POSBEL = NEGBEL = \emptyset$,
$NTRBEL = \{P(b), Q(a)\}$, $GRPBEL = \{P(a), Q(b)\}$. □

In Example 3.3 each of the ground literals of S occurs in exactly one half of the models, but there is a significant difference between atoms of *GRPBEL* and *NTRBEL*. The atoms of *NTRBEL* may assume any value *independently* of beliefs about the rest of atoms of S. Indeed, suppose that a set of beliefs is consistent with S, and forms a model μ containing, say, $P(b) \in NTRBEL$. Since $P(b)$ occurs in no kernel, there exists, by Definition 3.1, a model $\mu' = \bar{\mu}_{P(b)}$. So, whatever our other beliefs are, assuming $P(b)$ true or false is equally justified and consistent with S. Consider now an atom of *GRPBEL*, e.g., $P(a)$. Since $P(a) \notin POSFCT \cup NEGFCT$, either of $P(a)$, $\neg P(a)$ is consistent with S, but if we have already got certain beliefs regarding other atoms of *GRPBEL*, then there might be no freedom for assuming the value of $P(a)$. Indeed, in Example 3.3, if we believe that $Q(b)$ is true, then $P(a)$ is false, but if $Q(b)$ is assumed false, the $P(a)$ should be believed true. This mutual dependence suggests the name *group* beliefs. The following procedure computes beliefs for all atoms of *GRPBEL*:

Procedure ASSUME_GRPBEL;

1. Choose any kernel κ; denote by G the set of group beliefs occurring in κ;
2. For every atom $A \in G$ do
   if literal A occurs in κ then assume A is true, else (* i.e., $\neg A \in \kappa$ *) assume A is false;
3. For every atom $B \in (GRPBEL - G)$ either assume B is true or assume B is false.

Lemma 3.2. Any assumption produced by procedure *ASSUME_GRPBEL* is consistent with S.

Proof. By Definition 3.1, the kernel κ chosen by *ASSUME_GRPBEL* (step 1) represents a model containing all the assumptions for the atoms of *GRPBEL* produced by the procedure. □

Corollary 3.1. If every kernel of S contains at most one atom of *GRPBEL*, then for all atoms of *GRPBEL* any assumption (*true* or *false*) is consistent with S.

Proof. By Definition 3.2, for every atom $A \in GRPBEL$ there are kernels κ_1, κ_2 such that $A \in \kappa_1$, $\neg A \in \kappa_2$. Let T be any set of assumptions for all atoms of *GRPBEL*, and an atom A be true (or false) in T. By the premise, kernels κ_1, κ_2 contain no group beliefs except A, hence κ_1 (resp., κ_2) represents a model of S containing T. □

4 Plausible World Assumption (a semantic approach)

Consider a logic system S describing a real world. Some facts of the world are provable in S, but some of them are not, since S reflects our incomplete knowledge of the world. Concerning such undefined facts we would like to have at least certain beliefs about them that would represent a *plausible* image of the world. If the system describes the world adequately, then any specific realization of the world corresponds to a model of S. So, the more models of S are in accord with our beliefs, the more plausible is our conception of the world. With this intention, let us call a *Plausible World Assumption* (*PWA*) the following set of facts and beliefs.

Definition 4.1. (Together with Definition 3.2 provides a semantic definition of *PWA*).
(i) For all $A \in POSFCT$ assume A is true;
(ii) For all $A' \in NEGFCT$ assume A' is false;
(iii) For all $B \in POSBEL$ assume B is true;
(iv) For all $B' \in NEGBEL$ assume B' is false;
(v) For all $E \in NTRBEL$ either assume E is true or assume E is false;
(vi) For all $F \in GRPBEL$ assume the value produced by procedure *ASSUME_GRPBEL*. □

Lemma 4.1. Let $PWA(S)$ denote the set of truth values assumed due to the *PWA* for all facts and beliefs of S. Then $PWA(S)$ is a model of S.

Proof. Consider a kernel κ chosen by procedure *ASSUME_GRPBEL*. For all atoms A occurring in κ, if $A \in POSFCT$ then $A \in \kappa$, and due to Definition 4.1 A is assumed true, $PWA(A) = 'true'$, else if $A \in NEGFCT$ then $\neg A \in \kappa$, and $PWA(A) = 'false'$, else if $A \in POSBEL$ then $A \in \kappa$, and

$PWA(A) = 'true'$, else if $A \in NEGBEL$ then $\neg A \in \kappa$, and $PWA(A) = 'false'$, else if $A \in GRPBEL$ then by Lemma 3.2, $PWA(A)$ satisfies κ. Note that κ contains no atoms belonging to $NTRBEL$. So, $PWA(S)$ agrees with κ on all its literals, and hence defines a model of S represented by κ. □

$PWA(S)$ is not unique. Definition 4.1 determines a set of *plausible models* whose cardinality is at least $2^{|NTRBEL|}$. Lemmas 3.1, 3.2, 4.1 imply the following theorem.

Theorem 4.1. (a) *PWA* is consistent with Horn and non-Horn systems;

(b) *PWA* defines beliefs for all atoms of $HB(S)$ leaving no one undefined;

(c) For all atoms of *POSFCT*, *NEGFCT*, *POSBEL*, *NEGBEL*, *NTRBEL* the *PWA* assigns truth values that agree with as many models of the system as possible. □

Corollary 4.1. An assumption different from $PWA(S)$ may be inconsistent with S.

Proof. By Lemma 3.1, for all atoms $A \in POSFCT$ and $B \in NEGFCT$ assuming that A is false or B is true is inconsistent with S. The following example shows that violating Definition 4.1 (iii), (iv), (vi) may also lead to inconsistency. □

Example 4.1. $S = \{P(a) \vee \neg P(b),$
$\neg P(b) \vee \neg Q(a), P(a) \vee Q(a), P(b) \vee Q(a),$
$P(a) \vee \neg Q(b), P(b) \vee \neg Q(b), \neg Q(a) \vee \neg Q(b)\}$.
This system has two kernels:
$\kappa_1 = \{P(a), P(b), \neg Q(a)\}$, $\kappa_2 = \{\neg P(b), Q(a), \neg Q(b)\}$,
determining the following classification:

$POSFCT = NEGFCT = \emptyset$,
$POSBEL = \{P(a)\}$, $NEGBEL = \{Q(b)\}$,
$NTRBEL = \emptyset$, $GRPBEL = \{P(b), Q(a)\}$.

By Definition 4.1 there are two plausible assumptions (corresponding to the two possible choices of a kernel by procedure *ASSUME_GRPBEL*):

$PWA_1(S) = \{P(a), P(b), \neg Q(a), \neg Q(b)\}$,
$PWA_2(S) = \{P(a), \neg P(b), Q(a), \neg Q(b)\}$.

The following assumptions, different from $PWA_1(S)$, $PWA_2(S)$, are inconsistent with S:

$ass_1 = \{\neg P(a), P(b), \neg Q(a), \neg Q(b)\}$ violates Definition 4.1 (iii),

$ass_2 = \{P(a), \neg P(b), Q(a), Q(b)\}$ violates Definition 4.1 (iv),

$ass_3 = \{P(a), P(b), Q(a), \neg Q(b)\}$ violates Definition 4.1 (vi). □

Let us consider the relationship between classifications produced due to *PWA* and *GCWA* (displayed in Figure 1). By Theorem 3.1:

A literal A occurs in every kernel iff it occurs in every minimal model, hence, $POSFCT = PD$.

On the other hand, if literal A occurs in no kernel, then it belongs to no minimal model, so, $NEGFCT \cup NEGBEL \cup NTRBEL \subset ND$.

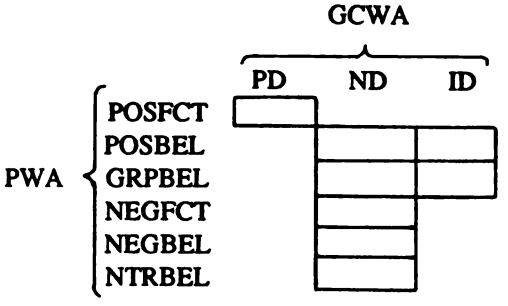


Figure 1.

If literal A occurs in some but not all kernels, then it may belong to no minimal model or to some but not all of them. Hence, $POSBEL \cup GRPBEL \subset ND \cup ID$.

Finally, if literal A belongs to some but not all minimal modes, then it occurs in some but not all kernels, hence, $ID \subset POSBEL \cup GRPBEL$.

5 Syntactic definition of PWA

Definitions 3.2, 4.1 present a model theoretic description of *PWA*. Now, following the line of [Minker, 1982], we give a proof theoretic definition of *PWA*, and show that both are equivalent.

Definition 5.1. (Syntactic definition of *PWA*) Consider a consistent system S, and an atom A. Let $S'(A)$ denote a set of all clauses $C \in S$ such that $S \nvdash (C - \{A, \neg A\})$. So, every clause of $S'(A)$ contains either A or $\neg A$. Then

(i) $A \in POSFCT$ iff $S \vdash A$;

(ii) $A \in NEGFCT$ iff $S \vdash \neg A$;

(iii) $A \in POSBEL$ iff $A \notin POSFCT$ and A is a *pure literal* in $S'(A)$, which means that literal A occurs in $S'(A)$, but $\neg A$ does not;

(iv) $A \in NEGBEL$ iff $A \notin NEGFCT$ and $\neg A$ is a pure literal in $S'(A)$;

(v) $A \in NTRBEL$ iff $S'(A) = \emptyset$;

(vi) $A \in GRPBEL$ iff both A and $\neg A$ occur in $S'(A)$. □

Theorem 5.1. The semantic and syntactic definitions of *PWA* are equivalent with regard to the

classification of atoms.

Proof. (i),(ii) $S \vdash A$ (or $S \vdash \neg A$) iff A (resp., $\neg A$) belongs to every model, and hence (by Lemma 3.1) to every kernel of S.

(iii) $\leftarrow$: Consider a clause $C \in S'(A)$ such that $C = A \vee D$. Since $S \nvdash D$, there exists a model μ of S falsifying D. Hence $A \in \mu$, but an interpretation $I = \bar{\mu}_A$ is not a model of S since I falsifies C. Therefore literal A must occur in a kernel representing μ. As $A \notin POSFCT$, there is a model η containing $\neg A$. Then $\eta' = \bar{\eta}_A$ is also a model. Indeed, η' satisfies every clause of $S'(A)$ (since A is a pure literal in $S'(A)$) and every clause $C' \in (S - S'(A))$ (since $S \vdash (C' - \{A, \neg A\})$). Therefore $\neg A$ occurs in no kernel representing η, and so in no kernel at all.

$\rightarrow$: Suppose A is not a pure literal in $S'(A)$ that contains clauses C_1, C_2 such that $C_1 = A \vee D_1$, $C_2 = \neg A \vee D_2$. Since $S \nvdash D_1$ and $S \nvdash D_2$, there exist models μ, η that falsify D_1, D_2, respectively, such that $A \in \mu$, $\neg A \in \eta$. But neither $I_1 = \bar{\mu}_A$ nor $I_2 = \bar{\eta}_A$ is a model of S, so A occurs in a kernel κ_1 representing μ, while $\neg A$ occurs in a kernel κ_2 representing η.

(iv) By the same argument as (iii).

(v) $\leftarrow$: If $S'(A) = \varnothing$ then for all clauses $C \in S$, $S \vdash (C - \{A, \neg A\})$. Hence, for every model μ of S, its conjugate w.r.t. A, $\bar{\mu}_A$, is also a model of S. This implies (by Lemma 3.1) that neither A nor $\neg A$ occurs in any kernel of S.

$\rightarrow$: If neither A nor $\neg A$ occurs in any kernel, then for every model μ of S, $\bar{\mu}_A$ is also a model. Suppose now that $S'(A) \neq \varnothing$, and consider a clause C of $S'(A)$ containing A or $\neg A$. As $S \nvdash (C - \{A, \neg A\})$, there exists a model η of S that falsifies $C - \{A, \neg A\}$. Hence $\bar{\eta}_A$ falsifies C, and so is not a model of S — a contradiction.

(vi) $\leftarrow$: Let both A and $\neg A$ occur in $S'(A)$, and consider clauses $C_1, C_2 \in S'(A)$ such that $C_1 = A \vee D_1$, $C_2 = \neg A \vee D_2$. By the argument used in (v), there exist models μ, η of S such that $A \in \mu$, $\neg A \in \eta$, and neither $\bar{\mu}_A$ nor $\bar{\eta}_A$ is a model of S. Hence, there must exist kernels κ_1, κ_2 such that $A \in \kappa_1$, $\neg A \in \kappa_2$.

$\rightarrow$: Suppose that (a) there are kernels κ_1, κ_2 such that $A \in \kappa_1$, $\neg A \in \kappa_2$, but (b) it is not true that both literals A and $\neg A$ occur in $S'(A)$. The latter means either that $S'(A) = \varnothing$ (which contradicts (a) by (v)), or that literal A (or $\neg A$) is pure in $S'(A)$, but this contradicts (a) by (i)-(iv). □

Example 5.1. $S = \{P(a), \neg Q(b), P(x) \vee Q(x), \neg P(x) \vee Q(y) \vee \neg R(x,y), P(y) \vee T(y)\}$. Clearly, $P(a) \in POSFCT$, $Q(b) \in NEGFCT$. Let us classify other atoms:

Since $\{\neg Q(b), P(x) \vee Q(x), \neg P(b)\}$ is unsatisfiable, we have $S \vdash P(b)$, so $P(b) \in POSFCT$.

$S'(Q(a)) = \{\neg P(a) \vee Q(a) \vee \neg R(a,a), \neg P(b) \vee Q(a) \vee \neg R(b,a)\}$, hence $Q(a) \in POSBEL$.

$S'(R(a,a)) = \{\neg P(a) \vee Q(a) \vee \neg R(a,a)\}$, so $R(a,a) \in NEGBEL$.

$S \vdash \neg R(a,b)$ since $\{P(a), \neg Q(b), \neg P(a) \vee Q(b) \vee \neg R(a,b), R(a,b)\}$ is inconsistent, hence $R(a,b) \in NEGFCT$.

$S'(R(b,a)) = \{\neg P(b) \vee Q(a) \vee \neg R(b,a)\}$, so $R(b,a) \in NEGBEL$.

$S \vdash \neg R(b,b)$ since $\{\neg Q(b), P(b) \vee Q(b), \neg P(b) \vee Q(b) \vee \neg R(b,b), R(b,b)\}$ is inconsistent, so $R(b,b) \in NEGFCT$.

$S'(T(a)) = S'(T(b)) = \varnothing$, hence $T(a), T(b) \in NTRBEL$.

Thus, $POSFCT = \{P(a), P(b)\}$, $NEGFCT = \{Q(b), R(a,b), R(b,b)\}$, $POSBEL = \{Q(a)\}$, $NEGBEL = \{R(a,a), R(b,a)\}$, $NTRBEL = \{T(a), T(b)\}$, $GRPBEL = \varnothing$. □

6 Relative monotonicity

Consider a logic system S and a set F of all formulae consistent with S. We say that S is *monotonic* if for all formulae $A, B \in F$, $S \vdash A$ entails $S \cup \{B\} \vdash A$, where $\vdash$ is the provability relation. Otherwise S is *non-monotonic* [Bossu and Siegel, 1985, Gabbay, 1985, McCarthy, 1980, Minker, 1982, Reiter, 1980].

Pure first order logic systems are monotonic, while Negation as Failure [Clark, 1978], CWA [Reiter, 1978], GCWA [Minker, 1982] and its extensions [Gelfond *et al.*, 1986, Yahya and Henschen, 1985], Circumscription [McCarthy, 1980], Negation as Inconsistency [Gabbay and Sergot, 1986] are non-monotonic.

Let $S \vdash A$ mean that A is either a provable fact in S ($S \vdash A$) or a belief consistent with S, although not provable in S ($S \nvdash A$). Gabbay [1985] studied *restricted monotonicity* (*RM*) such that if $S \vdash A$ and $S \vdash B$ then $S \cup \{B\} \vdash A$. That is, under the *RM* if S is augmented by a formula *deducible* from S due to the relation $\vdash$, this does not affect deducibility of other formulae deducible from S.

We consider a more general type of non-monotonicity, *relative monotonicity*, that is characteris-

tic of *PWA*.

Definition 6.1. (Relative monotonicity). Let V, W be sets of formulae (not necessarily disjoint). If for all $A \in V$ and all $B \in W$ and $B \neq \neg A$, $S \vdash A$ entail $S \cup \{B\} \vdash A$, then V is *monotonic w.r.t.* W. □

If W is empty for all V, then S is totally non-monotonic, but many non-monotonic systems display certain degree of relative monotonicity. For instance, due to *RM* both V and W are the same set of all formulae deducible from S. Let $FACTS(S)$, $BELS(S)$, $CONS(S)$ denote, respectively, sets of truth values of all facts provable in S, all beliefs in S under certain assumption, all ground literals consistent with S. Then under Negation as Failure, CWA (for Horn systems), and GCWA (for Horn and non-Horn systems) $FACTS(S)$ is monotonic w.r.t. $CONS(S)$, and $FACTS(S) \cup BELS(S)$ is monotonic w.r.t. itself. However, $BELS(S)$ is *not* monotonic w.r.t. $CONS(S)$, which accounts for the impact of wrong beliefs on other ones in S.

Example 6.1. Consider the system S of Example 3.2, and its minimal model $\mu_0 = \{P(a)\}$. According to *GCWA*, $P(b)$, $Q(a)$, $Q(b)$ should be believed false, so $FACTS = \{P(a)\}$, $BELS = \{\neg P(b), \neg Q(a), \neg Q(b)\}$, $CONS = \{P(a), P(b), \neg P(b), Q(a), \neg Q(a), Q(b), \neg Q(b)\}$. Adding to S any literal of $CONS$ does not change the provability of $P(a)$, but if S is augmented, say, by $Q(a) \in CONS$ contradicting the initial belief, then the assumption that $P(b)$ is false must be changed in $BELS$. □

Suppose that under certain assumption a set V is monotonic w.r.t. a set W. Then the larger are V and W the less troubles are caused by discovering a wrong belief, and so the higher is the degree of relative monotonicity of the system. If $V = FACTS(S) \cup BELS(S)$ and $W = CONS(S)$ then S becomes monotonic. This is the case with first order logic systems, since there are no beliefs, $BELS(S) = \emptyset$. The following theorem shows that *PWA* possesses the property of relative monotonicity to a higher degree than the known approaches to systems with beliefs.

Theorem 6.1. Given a system S, the set of truth values assumed due to *PWA* for all atoms of $HB(S) - GRPBEL(S)$ is monotonic w.r.t. $CONS(S)$.

Proof. Suppose that S is augmented by a literal $L \in CONS(S)$ becoming $S_L = S \cup \{L\}$. Let $L \in \{A, \neg A\}$, and consider $PWA(S)$ and $PWA(S_L)$. We have to show that for all atoms of $HB(S) - (GRPBEL(S) \cup \{A\})$ their truth values in $PWA(S_L)$ agree with those in $PWA(S)$. Let $MOD(S)$, $KER(S)$ stand for sets of all models and kernels of S, respectively.

(i) Consider $A \in POSFCT(S)$, so $A \in CONS(S)$ and $\neg A \notin CONS(S)$. Then $MOD(S_A) = MOD(S)$, and so $KER(S_A) = KER(S)$. Hence $PWA(S_A) = PWA(S)$.

(ii) If $A \in NEGFCT(S)$ then by the same argument $PWA(S_{\neg A}) = PWA(S)$.

(iii) Denote by $MOD(S, L)$, $KER(S, L)$, respectively, the sets of models and kernels of S containing literal L. If $A \in POSBEL(S)$ then every model of S_A is a model of S, but a model μ of S is a model of S_A iff μ contains A. Hence $MOD(S_A) = MOD(S, A)$, and $KER(S_A) = \{\kappa \cup \{A\} \mid \kappa \in KER(S)\}$. This implies the following *PWA* classification of S_A relatively to that of S:

$$POSFCT(S_A) = POSFCT(S) \cup \{A\},$$
$$NEGFCT(S_A) = NEGFCT(S),$$
$$POSBEL(S_A) = POSBEL(S) - \{A\},$$
$$NEGBEL(S_A) = NEGBEL(S),$$
$$NTRBEL(S_A) = NTRBEL(S).$$

Now suppose that the initial belief that an atom $A \in POSBEL(S)$ is true, turned out to be wrong, and so S must be augmented by $\{\neg A\}$, becoming $S_{\neg A}$. Then $MOD(S_{\neg A}) = MOD(S, \neg A)$, and $KER(S_A) = \{\kappa \cup \{\neg A\} \mid \kappa \in KER(S)$ *and* $A \notin \kappa\}$. So, only those kernels of S which don't contain A (augmented in $S_{\neg A}$ by $\{\neg A\}$) constitute $KER(S_{\neg A})$. This implies the following classification:

The atom A, that has been a positive belief in S, becomes a negative fact in $S_{\neg A}$;

Every atom of $POSFCT(S)$ occurs in every kernel of $S_{\neg A}$, hence $POSFCT(S) \subseteq POSFCT(S_{\neg A})$;

By the same argument, $NEGFCT(S) \subseteq NEGFCT(S_{\neg A})$;

Consider an atom $B \in POSBEL(S)$; in $S_{\neg A}$ literal B may occur either in some but not all kernels, or in all kernels, or in no kernels, hence $POSBEL(S) \subseteq (POSBEL(S_{\neg A}) \cup POSFCT(S_{\neg A}) \cup NTRBEL(S_{\neg A}))$;

By the same argument, $NEGBEL(S) \subseteq (NEGBEL(S_{\neg A}) \cup NEGFCT(S_{\neg A}) \cup NTRBEL(S_{\neg A}))$;

No atom of $NTRBEL(S)$ occurs in a kernel of $S_{\neg A}$, hence $NTRBEL(S) \subseteq NTRBEL(S_{\neg A})$.

Thus, the initial assumptions for all atoms of $HB(S)$ except those of $GRPBEL(S) \cup \{A\}$ agree with $PWA(S_A)$ and $PWA(S_{\neg A})$. Concerning an atom of $GRPBEL(S)$, its value in $PWA(S)$ may differ from that in $PWA(S_{\neg A})$, e.g., if for $PWA(S)$ procedure *ASSUME_GRPBEL* used a kernel of $KER(S)$ that contains A, and so does not belong to $KER(S_{\neg A})$.

(iv) The same argument as in (iii) proves that for all atoms $E \in NEGBEL(S)$, if S is augmented by either $\{E\}$ or $\{\neg E\}$, this does not change the initial assumptions for all atoms of $HB(S)$ except those of $GRPBEL(S) \cup \{E\}$.

(v) Let an atom $A \in NTRBEL(S)$ be assumed true in $PWA(S)$, and consider S_A. All models of S containing A constitute $MOD(S_A)$. These models are represented in S by all kernels of $KER(S)$. In S_A atom A becomes a positive fact, so $KER(S_A) = \{\kappa \cup \{A\} \mid \kappa \in KER(S)\}$. Therefore $PWA(S_A) = PWA(S)$.

If $\{\neg A\}$ is added to S, then $MOD(S_{\neg A}) = MOD(S, \neg A)$, so $KER(S_{\neg A}) = \{\kappa \cup \{\neg A\} \mid \kappa \in KER(S)\}$. Hence, A becomes a negative fact in $S_{\neg A}$, while the assumptions for *all* other atoms remain unchanged.

By the same argument we reach the same conclusion regarding an addition to S of $\{B\}$ or $\{\neg B\}$ for any atom $B \in NTRBEL(S)$ that is believed false in $PWA(S)$.

(vi) Consider an atom $A \in GRPBEL(S)$ assumed true (or false) in $PWA(S)$. By the argument of (iii),(iv), adding to S either $\{A\}$ or $\{\neg A\}$ does not change the initial assumption for all atoms of S except the members of $GRPBEL(S)$. □

Theorem 6.1 implies the following corollary.

Corollary 6.1. If $GRPBEL(S) = \emptyset$ then S is *monotonic* under *PWA* in the sense that $PWA(S)$ is monotonic w.r.t. $CONS(S)$. □

Example 6.2.

$$S_0 = \{P(a) \vee \neg P(b), P(a) \vee Q(a)\};$$

$\kappa_{01} = \{P(a)\}$, $\kappa_{02} = \{\neg P(b), Q(a)\}$;
$POSBEL = \{P(a), Q(a)\}$, $NEGBEL = \{P(b)\}$,
$NTRBEL = \{Q(b)\}$ (empty classes are omitted);
$PWA(S_0) = \{P(a), \neg P(b), Q(a), Q(b)\}$.

$$S_1 = S_0 \cup \{\neg P(b)\}$$

$$= \{\neg P(b), P(a) \vee \neg P(b), P(a) \vee Q(a)\};$$

$\kappa_{11} = \{P(a), \neg P(b)\}$, $\kappa_{12} = \{\neg P(b), Q(a)\}$;
$NEGFCT = \{P(b)\}$, $POSBEL = \{P(a), Q(a)\}$,
$NTRBEL = \{Q(b)\}$;
$PWA(S_1) = \{P(a), \neg P(b), Q(a), Q(b)\} = PWA(S_0)$;

$$S_2 = S_0 \cup \{P(b)\}$$

$$= \{P(b), P(a) \vee \neg P(b), P(a) \vee Q(a)\};$$

$\kappa_{21} = \{P(a), P(b)\}$;
$POSFCT = \{P(a), P(b)\}$, $NTRBEL = \{Q(a), Q(b)\}$;
$PWA(S_2) = \{P(a), P(b), Q(a), Q(b)\}$, which differs from $PWA(S_0)$ only in $P(b)$.

$$S_3 = S_0 \cup \{\neg Q(b)\}$$

$$= \{\neg Q(b), P(a) \vee \neg P(b), P(a) \vee Q(a)\};$$

$\kappa_{31} = \{P(a), \neg Q(b)\}$, $\kappa_{32} = \{\neg P(b), Q(a), \neg Q(b)\}$;
$NEGFCT = \{Q(b)\}$, $POSBEL = \{P(a), Q(a)\}$,
$NEGBEL = \{P(b)\}$;
$PWA(S_3) = \{P(a), \neg P(b), Q(a), \neg Q(b)\}$, which differs from $PWA(S_0)$ only in $Q(b)$. □

7 Summary

Consider a logic system describing a part of the real world. Since our knowledge of the domain is, in general, incomplete, there is a good deal of questions that cannot be answered definitely in the system. Wishing to form certain opinion about these undefined facts we have no choice but to resort to guesses, assumptions, beliefs, that are more or less in accord with the rest of our knowledge. Doing so, we should not be surprised when the reality disproves some of our beliefs, and we have to change them, and correct consequences drawn from wrong beliefs. So, incorporating beliefs in a system makes it non-monotonic. It is only natural that we would like to make our beliefs vulnerable as little as possible, in the sense that (a) each belief is as likely to be in accord with the reality as possible, and (b) a change of a belief affects as few other beliefs as possible.

We present a way of producing such relatively stable beliefs, called *Plausible World Assumption*. PWA has certain advantageous features compared to well known approaches to non-monotonic systems, such as Negation as Failure, Closed World Assumption, Generalized Closed World Assumption.

PWA provides beliefs for *all* atoms of the Herbrand base of a given non-Horn system leaving no one of them undefined.

Under PWA most of the assumptions correspond to the *majority* of models of the system, and so have a good chance to be approved by the reality.

We study a notion of *relative monotonicity*, and show that under PWA a system possesses a high degree of this feature, in the sense that most of its beliefs are not affected by any change of other ones.

In a given system PWA can *coexist* with GCWA and Circumscription. For this purpose PWA should be applied only to those atoms which are neither governed by GCWA nor circumscribed. This approach may continue the line of Protected Circumscription [Minker and

Perlis, 1985].

Acknowledgments

Many thanks to Michael Kifer for thoughtful discussions. I am grateful to the anonymous referees for their helpful comments.

References

[Bidoit and Hull, 1986]
Bidoit, N., and Hull, R., Positivism vs. minimalism in deductive databases. *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, Cambridge, Mass., 1986, 123-132.

[Bossu and Siegel, 1985]
Bossu, G., and Siegel, P., Saturation, nonmonotonic reasoning and the closed world assumption. *Artificial Intelligence 25* (1985), 13-63.

[Chang and Lee, 1973]
Chang, C.-L., and Lee, R.C.-T., *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, New York, 1973.

[Clark, 1978]
Clark, K.L., Negation as failure. In *Logic and Data Bases*, Gallaire, H., and Minker, J. (eds.), Plenum Press, New York, 1978, 293-322.

[Gabbay, 1985]
Gabbay, D.M., Theoretical foundations for non-monotonic reasoning in expert systems. In *Logics and Models of Concurrent Systems*, K.R.Apt, ed., NATO ASI Series, vol. F13, Springer-Verlag, 1985, 439-455.

[Gabbay and Sergot, 1986]
Gabbay, D.M., and Sergot, M.J., Negation as inconsistency. *J. Logic Programming 5*, 1 (1986), 1-35.

[Gelfond *et al.*, 1986]
Gelfond, M., Przymusinska, H., and Przymusinski, T., The extended closed world assumption and its relation to parallel circumscription. In *Proc, 5th ACM SIGACT SIGMOD Symp. on Principles of Database Systems*, 1986, 133-139.

[Horn, 1951]
Horn, A., On sentences which are true of direct unions of algebras. *J. Symb. Logic 16* (1951), 14-21.

[McCarthy, 1980]
McCarthy, J., Circumscription - a form of non-monotonic reasoning. *Artificial Intelligence 13* (1980), 27-39.

[Minker, 1982]
Minker, J., On indefinite databases and the closed world assumption. In *Lecture Notes in Computer Science 138*, Springer- Verlag, 1982, 292-308.

[Minker and Perlis, 1985]
Minker, J., and Perlis, D., Computing protected circumscription. *J. Logic Programming 4* (1985), 235-249.

[Przymusinski, 1986]
Przymusinski, T.C., On the semantics of stratified deductive databases. In *Proc. Workshop on Foundations of Deductive Databases and Logic Programming*, Minker, J. (ed.), Washington, D.C., 1986, 433-443.

[Reiter, 1978]
Reiter, R., On closed world data bases. In *Logic and Data Bases*, Gallaire, H., and Minker, J. (eds.), Plenum Press, New York, 1978, 55-76.

[Reiter, 1980]
Reiter, R., A logic for default reasoning. *Artificial Intelligence 13* (1980), 81-132.

[Yahya and Henschen, 1985]
Yahya, A., and Henschen, L.J., Deduction in non-Horn databases. *J. of Automated Reasoning, 1* (1985), 141-160.

Relating autoepistemic and default logics

Wiktor Marek *
Mirosław Truszczyński *
Department of Computer Science
University of Kentucky
Lexington, KY 40506-0027

Abstract

We investigate the relationship between autoepistemic logic and default logic. Our approach is syntactic in nature. In each logic we find three classes of objects – minimal sets closed under defaults, weak extensions, extensions for default logic, and minimal stable theories, expansions and robust expansions for autoepistemic logic – so that for a default theory (D,W), E is a minimal set closed under defaults (resp. weak extension, extension) if and only if E is the objective part of a minimal stable theory (resp. expansion, robust expansion) for the autoepistemic interpretation of (D,W). Similar results for the converse direction hold only in the case of minimal stable sets and minimal sets closed under defaults, and expansions and weak extensions. A weaker result holds for robust expansions and extensions. This multilevel correspondence between default and autoepistemic logics pinpoints the exact character of the equivalence of their expressive powers.

1 Introduction

In this paper we investigate connections between *default* and *autoepistemic* logics. Both of these logics are nonmonotonic and received considerable attention lately as they capture important aspects of commonsense reasoning ([Reiter, 1980, Moore, 1985, Marek and Truszczyński, 1988, Konolige, 1988a]).

Default logic uses the language of classical logic. It deals with *default theories* – pairs (D,W), where W is a collection of formulas of a first order language (initial knowledge) and D is a collection of *defaults*. Defaults are *context-sensitive* inference rules. They allow derivations of new facts out of previously proven ones in the presence of an appropriate context. The context-sensitivity of these rules results in nonmonotonicity. Having more facts as initial assumptions may result in forcing us to consider contexts that block applicability of previously used rules. The context-sensitivity of the derivation process results (in some cases) in generation of many possible sets of consequences of a default theory.

Autoepistemic logic is a modal logic with a single modal operator K. The intended interpretation of $K\phi$ is "ϕ is known to a fully introspective agent". This interpretation results in some natural requirements on the total knowledge of the agent (that is the set of all formulas known to the agent). Reasoning in autoepistemic logic is also context-sensitive, and there are (in some cases) many candidates for the agent's knowledge.

It was believed that the expressive powers of default and autoepistemic logics are similar [Reiter, 1987] and an important attempt to capture and formalize this similarity was undertaken by Konolige [1988a]. His line of argument can be described as follows. The central role in both logics is played by certain theories referred to as extensions (in default logic) and expansions (in autoepistemic logic). What one needs is an embedding of default theories into autoepistemic logic that would establish a correspondence between extensions of a default theory (D,W) and expansions of the autoepistemic image of (D,W). Konolige found a natural embedding (denoted in our paper tr_K) with the property that extensions of (D,W) are *objective parts* of expansions of the corresponding autoepistemic theory. He realized, though, that the converse implication does not hold, i.e., the objective part of an expansion of the translation of a default theory (D,W) need not to be an extension of (D,W). Consequently, extensions and expansions do not form a perfect match under the translation tr_K. At that point there are at least two possibilities for further studies. One is to strengthen the notion of expansion within the autoepistemic logic in order to eliminate those expansions, whose objective parts are not extensions, thus capturing the precise expressive power of extensions. This is the direction taken by Konolige. The other direction is to weaken, on the default side, the notion of extension to find a

*Work partially supported by National Science Foundation grant RII 8610671 and the Commonwealth of Kentucky EPSCoR program.

perfect match for a less expressive notion of expansion. Both directions are studied in this paper.

One should mention here that one of the most important benefits of having a correspondence between default logic and autoepistemic logic could be a discovery of some semantics for default logic. Autoepistemic logic possesses a very natural semantics called *list semantics*, introduced by Moore [1985] and studied by Konolige [1988a]. This semantics satisfies the completeness property, i.e., semantical notion of consequence has an exact syntactical counterpart. In default logic, as introduced by Reiter, consequence is defined syntactically and so far no complete semantics seems to exist.

The idea of Konolige [1988a] was to strengthen the notion of expansion by restricting semantical means used to define expansion. He introduces two stronger classes of expansions: *moderately grounded* and *strongly grounded*. One of his results claims that strongly grounded expansions correspond to extensions. In this paper, we show that strongly grounded expansions are not adequate to exactly capture and reflect properties of extensions. The intuitive explanation for this is the following. Defaults are used as inference rules and there is no explicit mechanism to put them together to obtain new defaults or new formulas. However, after defaults are translated into autoepistemic logic, they become ordinary autoepistemic formulas and, hence, may be used *together* to generate new facts, both epistemic and not involving knowledge operator K. An example and more detailed discussion is given in Section 3. Konolige defines his class of expansions by means of certain autoepistemic valuations that use *stable* sets (cf Stalnaker, [1980]) as modal indices (contexts). That approach does not prevent formulas that are images of defaults from interacting, and that is precisely why strongly grounded expansions are not adequate to capture the meaning of extensions.

In this paper, unlike Konolige [1988a], we exploit syntactical properties of extensions and expansions, and their similarities. Our point of departure is Reiter's operator $\Gamma(E)$. We find that $\Gamma(E)$ is a fixed point of a certain operator $R^{D,E}$ which is monotone and uses E as a parameter. The fixed point equation of Reiter becomes then a requirement that the parameter used in the construction of a fixed point be equal to the fixed point itself. The operator $R^{D,E}$ is proof-theoretical in its nature. It captures properties of the notion of a *strong default proof* using initial knowledge W, rules D *and* the context E. A theory E is then an extension if and only if it coincides with the set of formulas which possess a strong default proof (with E serving as the context).

Employing these proof-theoretic observations to autoepistemic logic, we define two new classes of expansions by simulating the process in which agent might be deriving (autoepistemic) conclusions from a given set of initial assumptions. These two classes turn out to be different from the classes of moderately and strongly grounded expansions of Konolige. We show that one of these classes, so called robust expansions, is a perfect match for the class of extensions if, on the autoepistemic side we restrict the attention to the theories in a certain normal form (with formulas where occurrences of K are not nested). So, in the context of extensions, robust expansions and autoepistemic theories in the normal form, expressive powers of default and autoepistemic logic coincide.

In our second result we take a different approach; instead of strengthening the notion of expansion, we weaken the notion of extension. This is done by mimicking the proof-theoretic method discussed above. We introduce the notion of a *weak default proof* (from initial knowledge W, using defaults from D and with context E). We define E to be a *weak extension* precisely when E coincides with the collection of formulas weakly provable with E serving as a context. We then show that weak extensions exactly correspond to expansions (without any restrictions on the form of autoepistemic theories involved).

In addition to these two correspondence results for default and autoepistemic logic, one relating extensions with robust expansions and the other relating weak extensions with expansions, we also show one more result of this type. Namely, under the translation of Konolige, minimal sets closed under defaults on the side of default logic correspond to minimal (but not in the sense of inclusion) stable theories on the side of autoepistemic logic (see Section 4.3).

We stress the fact that our approach is syntactical. In this we follow Reiter [1980] and Etherington [1988], as opposed to Konolige's semantical approach. Not only this extends our repertoire of means to study default and autoepistemic logics but it allows to eliminate the use of "nonclassical" proof systems (such as $K45$ used in [Konolige, 1988a]). We also believe that only syntactical approach may lead eventually to some automatization of procedures to manipulate default and autoepistemic theories.

It was observed in [Marek and Truszczyński, 1989] that default logic is connected to the so called stable semantics for general logic programs, and may be treated as its precursor. It may be of deeper interest to further investigate the close relationship between logic programming and default logic which for many years developed in a complete separation.

Our results support general view about similarity of default and autoepistemic logic by revealing correspondences between extensions and robust expansions and weak extensions and expansions. However, for the moment at least, it is unclear whether autoepistemic theories with nested occurrences of K can be adequately interpreted within the default logic by means of extensions alone.

The paper is organized as follows. In Section 2 we recall the basics of both default and autoepistemic logics presenting the proof-theoretic approach to the notion of default extension (Subsection 2.1), a compact overview of autoepistemic logic (Subsection 2.2) and connections between both systems of reasoning (Subsection 2.3). In Section 3 we introduce proof-based notions of expansions for autoepistemic logic and prove that one of these (so called *robust* expansions) mimics precisely the notion of extension in default logic. Section 4 deals again with correspondence problem for expansions, extensions, minimal sets closed under defaults and stable sets. The results of this section can be formulated as follows. In each logic (default and autoepistemic) we find three classes of objects – minimal sets closed under defaults, weak extensions, extensions for default logic, and minimal stable theories, expansions and robust expansions for autoepistemic logic – so that for a default theory (D, W), E is a minimal set closed under defaults (resp. weak extension, extension) for (D, W) if and only if E is the objective part of a minimal stable theory (resp. expansion, robust expansion) for $tr_K(D, W)$. Similar results for the converse direction hold only in the case of minimal stable sets and minimal sets closed under defaults, and expansions and weak extensions. A weaker result holds for robust expansions and extensions. Finally, we conclude the paper summarizing the results and indicating directions for further research.

2 Preliminaries

Throughout the paper we consider the language $\mathcal{L}$ of propositional logic (whenever we discuss default logic) and its extension $\mathcal{L}_K$ by the modal operator K (whenever we talk about the autoepistemic logic). We also use the following notation: for $T \subseteq \mathcal{L}_K$ we define: $KT = \{K\phi : \phi \in T\}$, $\neg T = \{\neg\phi : \phi \in T\}$, and $\overline{T} = \mathcal{L}_K \setminus T$.

2.1 Default logic

Formally, a default is a triple $< \alpha, L, \omega >$, where $\alpha, \omega \in \mathcal{L}$, and L is a finite subset of $\mathcal{L}$. Formula α is called the *prerequisite* of the default, ω is called the *conclusion*, and the formulas of L are called the *justifications*. Customarily, we write defaults as $\frac{\alpha:L}{\omega}$, or $\frac{\alpha:\beta_1,\ldots,\beta_n}{\omega}$, if $L = \{\beta_1, \ldots, \beta_n\}$. For a default d we often denote its prerequisite by $p(d)$, its set of justifications by $j(d)$ and its conclusion by $c(d)$. Defaults can be viewed as context-sensitive inference rules. Let $E \subseteq \mathcal{L}$. If $p(d)$ has been proved, and all justifications in $j(d)$ are "possible" (their negations are not in E), then we accept the conclusion $c(d)$. Clearly, it depends on the set E – the *context* – which formulas can be derived.

A *default theory* is a pair (D, W), where D is a collection of defaults and $W \subseteq \mathcal{L}$. Let E be a context (i.e. $E \subseteq \mathcal{L}$). By a *strong proof* of a formula ϕ from the theory W by means of defaults from D and with respect to the context E, we mean a finite sequence $\phi_1, \ldots, \phi_n$ such that $\phi_n = \phi$ and, for every $1 \leq i \leq n$,

(a) ϕ_i is an element of W or an axiom of logic, or

(b) there are $j, k < i$ such that $\phi_k = (\phi_j \Rightarrow \phi_i)$ (i.e. ϕ_i is derived from ϕ_j and ϕ_k by means of modus ponens), or

(c) there is $d \in D$ such that $p(d) = \phi_j$ for some $j < i$, $\phi_i = c(d)$, and for all $\beta \in j(d)$, $\neg\beta \notin E$.

Given (D, W) and E, let $Cn^{D,E}(W)$ be the collection of formulas possessing a strong proof from W by means of defaults from D and with respect to E. Then $Cn^{D,E}(W)$ is a *context-dependent* consequence operation. Now we define an operator $R^{D,E}$ (where D is a collection of defaults and E is a context) as follows:

$$R^{D,E}(S) = Cn(S \cup \{c(d) : d \in D, p(d) \in S, \neg j(d) \cap E = \emptyset\}).$$

Then, we iterate operator $R^{D,E}$ on W:

$$R_0^{D,E}(W) = Cn(W),$$

$$R_{n+1}^{D,E}(W) = R^{D,E}(R_n^{D,E}(W)),$$

for $n \geq 0$. Thus,

$$R_{n+1}^{D,E}(W) = Cn(R_n^{D,E} \cup \{c(d) : d \in D, p(d) \in R_n^{D,E}(W), \neg j(d) \cap E = \emptyset\}).$$

Finally, we put

$$R_\infty^{D,E}(W) = \bigcup_{n=0}^{\infty} R_n^{D,E}(W).$$

It is easy to see that the operator $R^{D,E}$ is monotone and finitary. Hence the set $R_\infty^{D,E}(W)$ is the least fixed-point of the operator $R^{D,E}$ over W. The connection of the operator $R_\infty^{D,E}$ and the notion of strong proof is given by the following theorem.

Theorem 2.1.1 $Cn^{D,E}(W) = R_\infty^{D,E}(W)$.

The central notion in default logic is that of *extension*. We say that a theory E is an extension of (D, W) if $R_\infty^{D,E}(W) = E$. Taking into account Theorem 2.1.1, we see that E is an extension of (D, W) if and only if E coincides with the set of formulas possessing a strong proof out of (D, W) with E itself serving as a context. The above definition and its proof-theoretic interpretation do not coincide with the original definition of Reiter [1980]. Let us recall that Reiter defined an extension as a fixed point of the (nonmonotonic) operator Γ, where $\Gamma(E)$ is defined as the smallest theory T such that:

(a) T is closed under consequence operation for $\mathcal{L}$,

(b) $W \subseteq T$,

(c) if $d \in D$, $p(d) \in T$ and $\neg j(d) \cap E = \emptyset$. then $c(d) \in T$.

Then, Reiter defines E to be an extension of (D, W) if and only if $\Gamma(E) = E$. Both definitions are equivalent. This follows from the theorem essentially proved by Reiter [1980].

Theorem 2.1.2 $\Gamma(E) = R_\infty^{D,E}(W)$.

This result, intuitively, says that extensions are (or at least may be treated as) proof-theoretic in nature. Thus, extensions in the default logic play the role analogous to that played by sets of consequences in classical logic. Note however that the notion of a strong proof is weaker than that of classical logic as it takes the context into account (condition (c) of the definition). Therefore, unlike in classical logic where each collection of facts determines a unique set of its consequences, a default theory may have exactly one extension, many extensions or no extensions at all ([Reiter, 1980]).

The notion of strong proof introduced above may be further relaxed, by use of conclusions of "generating defaults", i.e., defaults for which in addition to consistency of justifications, prerequisite belongs to E. Thus we allow here the context E to intervene not only on the negative (justification) side of the default but also on the positive (prerequisite) side as well. Specifically, a sequence $\phi_1, \ldots, \phi_n$ is a *weak proof* of ϕ if $\phi_n = \phi$ and, for every $1 \leq i \leq n$,

(a) ϕ_i is an element of W or an axiom of logic, or

(b) there are $j, k < i$ such that $\phi_k = (\phi_j \Rightarrow \phi_i)$, or

(c) there is $d \in D$ such that $p(d) \in E$, for all $\beta \in j(d)$, $\neg\beta \notin E$, and $\phi_i = c(d)$.

The difference between strong and weak proofs lies in conditions under which defaults are applied in the proof (condition (c) in both definitions). Unlike in the strong proof, where the prerequisite of a default has to be derived in order for the default to be applicable, in the weak proof it is sufficient that the prerequisite belongs to E. This allows for the curious situation in which the "proof" of a fact may depend on the fact itself! (a phenomenon common in autoepistemic logic).

We define $Cn_w^{D,E}(W)$ be the collection of formulas possessing a weak proof from W by means of defaults from D and with context E. Given a default theory $\Delta = (D, W)$ and an arbitrary theory $E \subseteq \mathcal{L}$, define the collection of *generating defaults* for E, $GD(\Delta, E)$ as $\{d : d \in D \wedge p(d) \in E \wedge \neg j(d) \cap E = \emptyset\}$. Also, for a collection of defaults D define $c(D) = \{c(d) : d \in D\}$. The operation $GD(\Delta, E)$ was considered by Reiter [1980] and Etherington [1988] but only in the situation when E was an extension of (D, W).

Following the concept of extension introduced above, we say that E is a *weak extension* of (D, W) if and only if $E = Cn_w^{D,E}(W)$. The relationship between weak extensions and the operation $GD(\Delta, \cdot)$ is given by the following theorem.

Theorem 2.1.3 *E is a weak extension of $\Delta = (D, W)$ if and only if*

$$E = Cn(W \cup c(GD(\Delta, E))).$$

Proof (outline). One needs to show how a usual (that is in sense of $\mathcal{L}$) proof of ϕ from $W \cup c(GD(\Delta, E))$ is converted to a weak proof of ϕ from (D, W) with the context E, and conversely. We leave the details to the reader. □

2.2 Autoepistemic logic

Autoepistemic logic is concerned with theories in $\mathcal{L}_K$. An intended interpretation of a formula $K\phi$ is "ϕ appears on the list of sentences accepted by the agent as known". Due to that interpretation, with the set of sentences serving as a means of interpreting the modal operator K, the central role in autoepistemic logic is played by *stable* theories ([Stalnaker, 1980]). A set $T \subseteq \mathcal{L}_K$ is stable if

(a) $T = Cn(T)$,

(b) If $\phi \in T$ then $K\phi \in T$,

(c) If $\phi \notin T$ then $\neg K\phi \in T$.

The *kernel* or *objective part* of a stable theory T is its trace on $\mathcal{L}$, that is $T \cap \mathcal{L}$. The following facts were proved in [Moore, 1985] ((a) and (b)) and in [Marek, 1986] ((c)) They will be used frequently throughout the paper.

Proposition 2.2.1 (a) *If T is stable then its kernel is closed under consequence in $\mathcal{L}$.*
(b) *T is uniquely determined by its kernel; that is if T_1 and T_2 are stable, $T_1 \cap \mathcal{L} = T_2 \cap \mathcal{L}$, then $T_1 = T_2$.*
(c) *There exists an operator $E : \mathcal{P}(\mathcal{L}) \mapsto \mathcal{P}(\mathcal{L}_K)$ such that, for every theory $S \subseteq \mathcal{L}$, $E(S)$ is the unique stable theory $T \subseteq \mathcal{L}_K$ such that $T \cap \mathcal{L} = Cn(S)$.*

An explicit definition of the operator E due to Marek [1986] is given below. Let $\mathcal{L}_{K,n}$ be the fragment of the language $\mathcal{L}_K$ consisting of formulas of K-depth at most n. (Thus $\mathcal{L} = \mathcal{L}_{K,0}$). For $S \subseteq \mathcal{L}$ define:

$$E_0(S) = Cn(S),$$

$$E_{n+1}(S) = \mathcal{L}_{K,n+1} \cap Cn(E_n(S) \cup KE_n(S) \cup \neg K(\mathcal{L}_{K,n} - E_n(S))),$$

and finally,

$$E(S) = \bigcup_{n=0}^{\infty} E_n(S).$$

The definitions of stable theories and of $E(S)$ are syntactical in nature and thus in the spirit of default logic. One has to notice, however, that the original development of autoepistemic logic was semantical and based on a novel semantics for the language $\mathcal{L}_K$ ([Moore, 1985]). This semantics can be described as follows. Let E be the collection of facts known to an agent.

In addition assume that the agent assigns the value 0 or 1 to elementary propositions (propositional variables) *not* involving the modal operator K; denote this assignment v. This generates the following semantics (called *list semantics*) for $\mathcal{L}_K$. The formulas *not* involving K are evaluated by means of v, the formulas of the form $K\phi$ which themselves are not within the scope of operator K are treated as atoms and are evaluated as follows: $K\phi$ is evaluated as 1 (*true*) if and only if ϕ belongs to E. "Mixed" formulas are evaluated according to usual conditions for satisfaction. We denote by $\models_{v,E} \phi$ the fact that ϕ evaluates to 1. Now, the notions of "E-tautology" and "E-entailment" follow. Namely, we define $\models_E \phi$ if and only if $\forall_v \models_{v,E} \phi$, and $I \models_E \phi$ if and only if $\forall_v(\models_{v,E} I \Rightarrow \models_{v,E} \phi)$.

The collection of formulas entailed by I (E is fixed) may serve as a, semantically defined, consequence of I. The crucial argument, due to Moore ([Moore, 1985]) goes as follows. Assume that $I \subseteq \mathcal{L}_K$ is given (think about I as *initial assumptions* of an agent, or "all agent knows" [Levesque, 1987]). Those lists E for which E coincides with the set of E-tautologies correctly represent agent's knowledge. Such lists satisfy the equality

$$E = \{\phi : I \models_E \phi\}.$$

and are called *expansions* of I.

The following proposition summerizes basic results concerning expansions proved in [Moore, 1985].

Proposition 2.2.2 (a) *E is an expansion of I if and only if*

$$E = Cn(I \cup KE \cup \neg K\overline{E}).$$

(b) *For every I, every expansion of I is stable.*
(c) *If E is stable then E is the only expansion of its kernel.*

Proposition 2.2.2 guarantees existence of unique expansions for theories $I \subseteq \mathcal{L}$. If, however, I contains non-objective formulas, I may have no expansion, exactly one expansion or many expansions. Examples are given in [Moore, 1985, Marek and Truszczyński, 1988, Konolige, 1988a] showing that all three general situations are possible.

As pointed by Moore, every formula ϕ of $\mathcal{L}_K$ is logically equivalent to a formula $\bigwedge \Theta_i$ where each Θ_i is of the form:

$$K\phi_1 \vee \ldots \vee K\phi_k \vee \neg K\psi_1 \vee \ldots \vee \neg K\psi_r \vee \omega.$$

Here $\omega \in \mathcal{L}$ and k or r or both can be equal 0. It follows that for every theory I there is a theory I^c consisting of implications of the form:

$$(\star) \qquad K\phi_1 \wedge \ldots \wedge K\phi_k \wedge \neg K\psi_1 \wedge \ldots \wedge \neg K\psi_r \Rightarrow \omega$$

with $\omega \in \mathcal{L}$, such that $Cn(I) = Cn(I^c)$. (We write such implications as $A \Rightarrow \omega$, possibly with indices). We shall call formulas of the form $(\star)$ *autoepistemic clauses* (ae-clauses for short). One easily shows that theories with the same consequences have also the same expansions. Thus we obtain the following result.

Proposition 2.2.3 *For every theory I there exists a theory I^c consisting of ae-clauses such that I^c has the same expansions as I.*

For a theory I consisting of ae-clauses, we define $W(I) = I \cap \mathcal{L}$ (formulas of I without any occurrence of K) and $D(I) = I \setminus W(I)$ (formulas of I with occurrences of K).

We say that a formula ϕ is *purely epistemic*, if for every atom p, every occurrence of p is within the scope of modal operator K.

Example 2.2.4 (a) *The formula $K(p \vee \neg K(r)) \wedge K(s)$ is purely epistemic,*
(b) *The formula $K(p \vee \neg K(r)) \wedge s$ is not purely epistemic.*

We say that ϕ is *stably equivalent* to ψ, (in symbols $\phi \sim_S \psi$) if for every stable theory T, $\phi \in T$ if and only if $\psi \in T$. Purely epistemic formulas have useful properties with respect to stable theories. The following facts are proved in [Marek and Truszczyński, 1988].

Proposition 2.2.5 (a) *If ϕ is purely epistemic then for every stable theory T, $\phi \in T$ or $(\neg\phi) \in T$.*
(b) *For every $\phi \in \mathcal{L}_K$ there exists a purely epistemic ψ such that $K-$depth of ψ is 1 and $\psi \sim_S \phi$.*

By Proposition 2.2.5 (a), $\phi \sim_S \psi$ implies that $\neg\phi \sim_S \neg\psi$, for *purely epistemic* ϕ and ψ. It is also easy to see that for every ϕ, $\psi \in \mathcal{L}_K$, $(K\phi \wedge K\psi) \sim_S K(\phi \wedge \psi)$. Thus, we may assume that the formula A in an ae-clause $A \Rightarrow \omega$, has at most one positive conjunct $K\phi$.

The following definition is closely related to Konolige's autoepistemic translation of default theories (see next subsection). A formula $\phi = K\alpha \wedge \neg K\phi_1 \wedge \ldots \wedge \neg K\phi_n \Rightarrow \omega$, where $\alpha, \omega, \phi_i \in \mathcal{L}$ is called an *ae-default* (positive part $K\alpha$, negative part $\neg K\phi_1 \wedge \ldots \wedge \neg K\phi_n$, or both may be missing). A theory I is said to be in the *normal* form if all formulas in I are ae-defaults. The next fact can be found both in [Konolige, 1988a] where it is proved by reference to logic $K45$ and in [Marek and Truszczyński, 1988] where it is proved semantically.

Proposition 2.2.6 *For every theory I there exists a theory I^+ (effectively constructible) consisting of ae-defaults such that I and I^+ possess the same expansions and the same stable sets.*

2.3 Connections between default and autoepistemic logics.

With the interpretation of $K\phi$ as "ϕ is known" in mind, a natural translation of a default $d = \frac{\alpha:\beta_1,\ldots,\beta_n}{\omega}$ into autoepistemic logic (due to Konolige [1988a]) is:

$$tr_K(d) = K\alpha \wedge \neg K\neg\beta_1 \wedge \ldots \wedge \neg K\neg\beta_n \Rightarrow \omega,$$

as $\neg K\neg\beta$ ($\neg\beta$ is "not known") can be interpreted as "β is possible". A default theory (D, W) is translated into an autoepistemic theory as follows:

$$tr_K(D, W) = W \cup \{tr_K(d) : d \in D\}.$$

Notice that $tr_K(D,W)$ is in the normal form (consists of ae-defaults). Konolige [1988a] points out that under this translation expansions do not correspond to extensions. The notion of expansion is too weak, as it allows context (modal index) T to be used both for generating $K\phi$ if $\phi \in T$ and $\neg K\phi$ if $\phi \notin T$, whereas in the definition of extension, the context is used only for the justification part of defaults (that is corresponds to formulas $\neg K\phi$ in the translation) and the prerequisite part (that corresponds to $K\psi$ in the translation) has to be *derived* earlier. Formally, Konolige shows the following fact.

Proposition 2.3.1 *If S is an extension of (D,W) then $E(S)$ is an expansion of $tr_K(D,W)$. Converse implication, however, does not hold.*

It is easy to give an example showing the second part of the proposition Consider $I = \{Kp \Rightarrow p\}$, possessing two expansions, $E(TAUT)$ and $E(Cn(p))$. But, $I = tr_K(D,W)$ with $W = \emptyset$ and $D = \{\frac{p:}{p}\}$, and theory (D,W) possesses only one extension, $TAUT$.

In order to find within autoepistemic logic an object exactly corresponding to the notion of extension Konolige strengthens the notion of expansion. A starting point for his argument is an observation that T is an expansion of $I \subseteq \mathcal{L}_K$ if and only if

$$T = \{\phi : I \cup KT_0 \cup \neg K(\mathcal{L} - T_0) \models_{SS} \phi\},$$

where $T_0 = T \cap \mathcal{L}$, and $\models_{SS} \phi$ denotes the fact that $\models_E \phi$ for every stable E. Modifying that fixed point equation Konolige gets two stronger classes of expansions. A set $T \subseteq \mathcal{L}_K$ is *moderately grounded* in I if

$$T = \{\phi : I \cup KI \cup \neg K(\mathcal{L} - T) \models_{SS} \phi\}.$$

A theory $T \subseteq \mathcal{L}_K$ is *strongly grounded* in a normal form set $I \subseteq \mathcal{L}_K$ ($I = \{A_i \Rightarrow \omega_i : 1 \leq i \leq k\}$) if

$$T = \{\phi : I' \cup KI' \cup \neg K(\mathcal{L} - T) \models_{SS} \phi\},$$

where $I' = \{A_i \Rightarrow \omega_i : \omega_i \in T\}$. Konolige shows that theories moderately grounded in I and strongly grounded in I (providing I is in normal form) are expansions of I. We refer to them as moderately and strongly grounded expansions, respectively. Konolige [1988a] claims that under the translation tr_K, extensions correspond precisely to strongly grounded expansions, that is, for every default theory (D,W), and every $S \subseteq \mathcal{L}$, S is an extension of (D,W) if and only if $E(S)$ is a strongly grounded expansion of $tr_K(D,W)$. This claim is, however, wrong. Consider the default theory $(\{\frac{p:}{p}, \frac{:\neg p}{p}\}, \emptyset)$. It is easy to see that it has no extensions. The autoepistemic translation of that theory is $I = \{Kp \Rightarrow p, \neg Kp \Rightarrow p\}$, and it possesses one strongly grounded expansion, namely $E(\{p\})$. The reason why strongly grounded expansions do not correspond to extensions is that defaults work as inference rules, hence, the effect of every single default is independent of all the other defaults. On the other hand, after defaults are translated into autoepistemic logic, they become formulas and definitions of expansions, moderately grounded expansions and strongly grounded expansions allow for interactions between them. *Those interactions are not necessarily expressible in terms of default proofs as is precisely the case in our example.* In our example p cannot be derived if $Kp \Rightarrow p$ or $\neg Kp \Rightarrow p$ are used separately, but p obviously is a consequence of the two formulas together. Here lies, in our opinion, the most significant difference between defaults and autoepistemic logics. Defaults work as inference rules and their prerequisites and justifications do not interplay as they are capable to do in the autoepistemic framework. Konolige tried to eliminate unwanted interactions by eliminating some formulas from I in the definition of strongly grounded expansion. As our example shows that was not enough. Later, Konolige [1988b] introduced one more type of an expansion by replacing set I' in the definition of strongly grounded expansion by set I'' defined to consist of these formulas $A_i \Rightarrow \omega_i$ from I, where $A_i = K\phi_1 \wedge \ldots \wedge K\phi_k \wedge \neg K\psi_1 \wedge \ldots \wedge \neg K\psi_r$, for which $\omega \in T$ *and* $\psi_1 \notin T, \ldots, \psi_r \notin T$. It turns out that expansions of that kind (call them *supergrounded* in I) exactly correspond to extensions.

Our results extend the work of Konolige. We investigate connections between default and autoepistemic logics on several levels. One of them deals with correspondence between extensions and appropriately defined expansions. We use a simpler, syntactical approach and eliminate the need for nonclassical proof systems like $K45$ used by Konolige. In the next two sections we define new classes of expansions by mimicing the process in which extensions are generated. One of them constitutes a perfect match for extensions. Thus, this class of expansions coincides with the class of supergrounded expansions of Konolige (but the definitions are different; our definition relies on properties of generating defaults). We also discuss relationship between weak extensions and expansions, and between minimal sets closed under defaults and minimal stable sets.

3 Proof-based expansions of autoepistemic theories

As mentioned above, Konolige [1988a] defines classes of expansions by means of their semantic properties; he refers to the satisfaction relation $\models_{v,T}$ and associated entailment relations. Here we take a different approach. Following ideas from logic programming, reflected in the area of default logic by Reiter's operator $R^{D,E}$ (as an alternative to the original Reiter's operator Γ), we apply the "operator" method within autoepistemic logic. As a result, we get two operators, A^T and B^T. It turns out that operator A^T does not, in general, eliminate the side effects which led

Konolige to formulate the notion of supergroundedness (however, with appropriate modifications, it does – this is the subject of next section). Our second operator B^T is technically more complex and less elegant but it faithfully simulates constructions from default logic.

Within autoepistemic logic we have a weak notion of provability inherited from classical logic in which every formula of the form $K\phi$ is treated as an atom. This notion of a proof is very restrictive, in particular a proof of formula ϕ does not allow to derive $K\phi$. This problem can be easily solved by adopting the necessitation rule. The real problem is how to incorporate context into an autoepistemic proof so that axiom (c) of stability is properly accounted for. Here, we accept (in the case of both A^T and B^T) a method inherited from default logic. We introduce a parameter T (context) and add new axioms – all formulas of the form $\neg K\phi$ for $\phi \notin T$. Thus, proofs become context dependent. This incorporation of a context is formalized below. An operator A is defined by

$$A(S) = Cn(S \cup KS).$$

Intuitively, operator A corresponds to the provability in modal logic which has both modus ponens and necessitation rule but no axioms for simplifications of modalities. The logical axioms in this logic are just substitutions of tautologies. The operator A is monotone and finitizable. By appropriate choice of extralogical axioms (in our case, formulas of the form $\neg K\phi$ for $\phi \notin T$) we get the desired nonmonotonicity. We iterate operator A, starting with the theory $I \cup KW(I) \cup \neg K\overline{T}$. It is precisely here that elements of context dependence and nonmonotonicity appear. The context intervenes in negative fashion at the beginning of the construction; subsequently only a monotone operator is applied. Let us define, for $I \subseteq \mathcal{L}_K$ (initial assumptions) and theory $T \subseteq \mathcal{L}_K$ (context)

$$A_0^T(I) = Cn(I \cup KW(I) \cup \neg K\overline{T}),$$

$$A_n^T(I) = A(A_{n-1}^T(I)) = Cn(A_{n-1}^T(I) \cup KA_{n-1}^T(I))$$

for $n > 0$, and let

$$A^T(I) = \bigcup_{n=0}^{\infty} A_n^T(I).$$

It is easy to see that A^T is a fixed point of operator A and in fact the least fixed point of A above $I \cup \neg K\overline{T}$. Hence the collection $A^T(I)$ consists precisely of the formulas which are provable from I *and* $\neg K\overline{T}$ (here the context is used in a negative fashion) using classical logic and the necessitation rule. The simplicity of the construction (it uses rather elementary modal logic and applies it to the initial knowledge I extended by some negative facts provided by the context) is rather surprising here, especially in view of Konolige's application of the logic $K45$. The adequacy of this construction follows from the following theorem. Before we state and prove it we introduce an additional technical assumption that simplifies our reasoning without any loss of generality.

Remark 3.1 Let (D, W) be a default theory. Define D' to be the set of defaults obtained from D by removing all trivial defaults of the form $\frac{:}{\omega}$, and define W' be the set of formulas obtained from W by adding formulas ω, for each default $\frac{:}{\omega}$ removed from D. It is evident that theories (D, W) and (D', W') have the same extensions. Moreover, they have the same autoepistemic translations ($tr_K(D, W) = tr_K(D', W')$). Therefore, throughout the rest of the section we consider only theories (D, W) such that for no $\omega \in \mathcal{L}$, $\frac{:}{\omega} \in D$. Recall that for $I \subseteq \mathcal{L}_K$, $W(I)$ consists of all formulas of I that do not involve operator K, and $D(I)$ consists of all other formulas from I. Due to our restriction on the default theories considered, we have that $W(I) = W$, for $I = tr_K(D, W)$, or equivalently, that $W(tr_K(D, W)) = W$.

Theorem 3.2 *If $A^T(I) = T$, then T is an expansion of I.*

Proof: We need to prove that if $A^T(I) = T$, then $T = Cn(I \cup KT \cup \neg K\overline{T})$. Now, $I \cup \neg K\overline{T} \subseteq A_0^T(I) \subseteq A^T(I) = T$. Moreover, if $\phi \in T$ then, for some n, $\phi \in A_n^T(I)$. Consequently, $K\phi \in A_{n+1}^T(I) \subseteq T$. This implies

$$I \cup KT \cup \neg K\overline{T} \subseteq T.$$

Since T is closed under Cn (as the union of an ascending chain of theories closed under Cn), it follows that

$$Cn(I \cup KT \cup \neg K\overline{T}) \subseteq T.$$

To show the converse inclusion we proceed by induction to prove that for all $n \in \mathcal{N}$, $A_n^T(I) \subseteq Cn(I \cup KT \cup \neg K\overline{T})$. This is certainly true for $n = 0$, as $A_0^T(I) = Cn(I \cup KW(I) \cup \neg K\overline{T})$, (since $I \subseteq T$, we have $KW(I) \subseteq KI \subseteq KT$; hence, $I \cup KW(I) \cup \neg K\overline{T} \subseteq Cn(I \cup KT \cup \neg K\overline{T})$). Assume $A_n^T(I) \subseteq Cn(I \cup KT \cup \neg K\overline{T})$. Note that since $A_n^T(I) \subseteq T$, we have $KA_n^T(I) \subseteq KT$. Hence,

$$Cn(A_n^T(I) \cup KA_n^T(I)) \subseteq Cn(I \cup KT \cup \neg K\overline{T}),$$

i.e., $A_{n+1}^T(I) \subseteq Cn(I \cup KT \cup \neg K\overline{T})$. □

An expansion T of I such that $A^T(I) = T$ is called an *iterative* expansion. Not every expansion is iterative. The following example is due to Konolige. For $I = \{Kp \Rightarrow p\}$ the expansion $E(Cn(p))$ is not iterative; adding formulas $\neg K\phi$ for ϕ *not in* $E(Cn(p))$ does not allow for derivation of p itself.

Although expansions generated by operator A^T from the initial knowledge mimic fairly well the way

in which strong default proofs work, they are not good enough; the unwelcome side effects persist. The example of $I = \{Kp \Rightarrow p, \neg Kp \Rightarrow p\}$ is again an indication of what happens here. The fact that we allow the formulas of I to interplay allows us to derive p regardless of the context.

Hence we are faced with this alternative: either we strengthen the operator, finding the means to eliminate the interplay between the formulas of I, or we do not change the operator, but instead modify collection I in order to eliminate the interplay. In fact both of these avenues are reasonable and both lead to the desired result – the reconstruction of the notion of default extension within the autoepistemic realm. In the remainder of this section we shall describe the first possibility. Our construction is, unfortunately, technically complicated. We introduce a new operator B^T. This operator is tailored to capture precisely the proof procedure of default logic. Because of this, we shall consider now theories consisting of ae-defaults only (normal theories), as such theories are images (under translation tr_K) of default theories.

Define, for $S \subseteq \mathcal{L}$,

$$B^{I,T}(S) = \mathcal{L} \cap \bigcup_{\phi \in D(I)} Cn(\{\phi\} \cup S \cup KS \cup \neg K\overline{T}),$$

and iterate this operator over $W(I)$:

$$B_0^T(I) = Cn(W(I)),$$
$$B_{n+1}^T(I) = B^{I,T}(B_n^T(I)),$$
$$B_\infty^T(I) = \bigcup_{n \in \mathcal{N}} B_n^T(I).$$

Finally, put

$$B^T(I) = E(B_\infty^T(I)).$$

The definition of the operator B^T is complicated and requires an explanation. The idea is to force that the formulas of $D(I)$ are used separately, thus not allowing any direct interplay. In addition, after each stage of the construction we leave only the objective information obtained during that stage and use that information only (possibly with a modal operator applied) later in the process. A weak form of an interplay still remains and has several sources. Firstly, if a formula $\phi \in \mathcal{L}$ is derived at some stage, the formula itself or $K\phi$ will appear at future stages and will interplay with other formulas from $D(I)$. Secondly, the negative knowledge of the form $\neg K\phi$, for $\phi \notin T$, is input into the construction at the very first stage at once. However, two formulas of $D(I)$ are never used together in the same derivation. Their objective consequences interplay but not the formulas themselves!

The adequacy of this construction is justified by the two main results of this section (Theorems 3.3 and 3.6). First of them shows that operator B^T is a refinement of operator A^T.

Theorem 3.3 *If $B^T(I) = T$, then T is an iterative expansion of I, that is $A^T(I) = T$. In particular, T is an expansion of I.*

An expansion T of I such that $T = B^T(I)$ is called *robust*. Later in the section, we shall prove that robust expansions exactly correspond to extensions. Robustness is a property of representation. The fact that T is a robust expansion of I does not imply that T is a robust expansion of I^+ for a theory I^+ such that $Cn(I) = Cn(I^+)$. This is certainly an unpleasant phenomenon. To see how this can happen, let us look at our standard example $I = \{\neg Kp \Rightarrow p, Kp \Rightarrow p\}$. The theory $I^+ = \{p\}$ which is logically equivalent to I possesses a robust expansion, namely $E(Cn(p))$. Yet I has no robust expansion, as can be easily verified. The fact that the theory $\{p\}$ possesses a robust expansion is no accident. In fact we have the following general fact.

Proposition 3.4 *If $S \subseteq \mathcal{L}$ then the only expansion $E(Cn(S))$ of S is robust.*

In order to prove Theorem 3.3 we need an auxiliary lemma.

Lemma 3.5 *If $B^T(I) = T$, then $I \subseteq T$.*

Proof: Since $I = W(I) \cup D(I)$, we need to prove that $W(I) \subseteq T$ and that $D(I) \subseteq T$. The first inclusion is obvious ($W(I) \subseteq B_0^T(I) \subseteq B_\infty^T(I) \subseteq B^T(I) = T$).

Now, let $\phi \in D(I)$. Then ϕ is of the form

$$K\alpha \wedge \neg K\beta_1 \wedge \ldots \wedge \neg K\beta_n \Rightarrow \gamma.$$

where α, β_i and $\gamma \in \mathcal{L}$, and part $K\alpha$ or negative conjuncts may be missing.

(a) If $\alpha \notin T$ then, since T is stable (as an expansion of $B_\infty^T(I)$), $\neg K\alpha \in B^T(I) = T$. Hence $\phi \in T$.

(b) If for some $j \leq n$, $\beta_j \in T$ then $K\beta_j \in T$ (by stability of T) and so, again, $\phi \in T$.

(c) Finally, assume that $\alpha \in T$ and for all $j \leq n$, $\beta_j \notin T$. Then, as $\alpha \in T = E(B_\infty^T(I))$, and since $B_\infty^T(I)$ is closed under consequence, we have $\alpha \in B_\infty^T(I)$ (by Proposition 2.2.1). Therefore, for some $n \in \mathcal{N}$, $\alpha \in B_n^T(I)$. But then $\phi, K\alpha, \neg K\beta_1, \ldots, \neg K\beta_n$ belong to

$$Cn(\{\phi\} \cup B_n^T(I) \cup KB_n^T(I) \cup \neg K\overline{T}).$$

In this set γ can be derived, and since γ belongs to $\mathcal{L}$, $\gamma \in B_{n+1}^T(I) \subseteq B_\infty^T(I) \subseteq T$. But since $\gamma \in T$, $\phi = K\alpha \wedge \neg K\beta_1 \wedge \ldots \wedge \neg K\beta_n \Rightarrow \gamma$ belongs to T as well, as T is closed under consequence. □

Proof of Theorem 3.3. Assume $T = B^T(I)$ and denote $S = B_\infty^T(I)$. Hence $T = E(S)$, in particular, T is stable. First, we prove by induction that for every n, $A_n^T(I) \subseteq T$. For $n = 0$ we have $A_0^T(I) = Cn(I \cup KW(I) \cup \neg K\overline{T})$. By Lemma 3.5, $I \subseteq T$. Hence $W(I) \subseteq T$ and, by stability, $KW(I) \cup \neg K\overline{T} \subseteq T$. Since T is closed under consequence, $A_0^T(I) \subseteq T$ follows. The induction step is an easy consequence of

stability of T, we omit details. Now, since $A^T(I) = \bigcup_{n=1}^{\infty} A_n^T(I)$, we obtain $A^T(I) \subseteq T$.

To complete the proof we need to show the converse inclusion $T \subseteq A^T(I)$. We recall that $T = E(S)$ and our first step is to prove that $S \subseteq A^T(I)$. To this end, we show by induction that $B_n^T(I) \subseteq A_n^T(I)$. For $n = 0$ we have $B_0^T(I) = Cn(W(I)) \subseteq Cn(I) \subseteq A_0^T(I)$. Assume now that $B_n^T(I) \subseteq A_n^T(I)$. Since $I \cup \neg K\overline{T} \subseteq A_0^T$ and $\{A_n^T(I)\}_{n\in N}$ is an ascending chain,

$$A_{n+1}^T(I) = Cn(I \cup A_n^T(I) \cup KA_n^T(I) \cup \neg K\overline{T}).$$

Hence, we have

$$Cn(\{\phi\} \cup B_n^T(I) \cup KB_n^T(I) \cup \neg K\overline{T}) \subseteq$$
$$Cn(I \cup A_n^T(I) \cup KA_n^T(I) \cup \neg K\overline{T}) = A_{n+1}^T(I),$$

for every $\phi \in D(I)$. This implies that $B_{n+1}^T(I) \subseteq A_{n+1}^T(I)$ and completes our inductive argument. Now,

$$S = \bigcup_{n=1}^{\infty} B_n^T(I) \subseteq \bigcup_{n=1}^{\infty} A_n^T(I) = A^T(I).$$

Next, we prove that $E(S) \subseteq A^T(I)$. This is done by induction again. We show that for every n, $E_n(S) \subseteq A^T(I)$. Since $E_0(S) = Cn(S) = S \subseteq A^T(I)$ (as proved above), the base for the induction is established. So, assume that $E_n(S) \subseteq A^T(I)$ and recall that

$$E_{n+1}(S) = \mathcal{L}_{K,n+1} \cap$$
$$Cn(E_n(S) \cup KE_n(S) \cup \neg K(\mathcal{L}_{K,n} - E_n(S)),$$

It follows easily from the definition that $A^T(I)$ is closed under the necessitation rule, i.e., whenever $\phi \in A^T(I)$ then also $K\phi \in A^T(I)$. This shows that out of three groups of generators for $E_{n+1}(S)$ first two are included in $A^T(I)$. Now, if $\phi \in \mathcal{L}_{K,n} - E_n(S)$ then $\phi \notin E(S) = T$. This follows from the fact established in [Marek, 1986] asserting that

$$E_n(S) \cap \mathcal{L}_{K,n} = E(S) \cap \mathcal{L}_{K,n}$$

Hence $\neg K\phi \in \neg K\overline{T}$. Now, $\neg K\overline{T} \subseteq A_0^T(I)$, hence $\neg K\phi \subseteq A^T(I)$. Taking all these facts into account, we find that

$$Cn(E_n(S) \cup KE_n(S) \cup \neg K(\mathcal{L}_{K,n} - E_n(S)) \subseteq A^T(I).$$

Consequently, $E_{n+1}(S) \subseteq A^T(I)$.

Thus, $E(S) = \bigcup_{n=1}^{\infty} E_n(S) \subseteq A^T(I)$. Hence, $T = A^T(I)$ and T is an iterative expansion of I. □

The next theorem gives the main result of this section. It shows that robust expansions exactly correspond to extensions.

Theorem 3.6 *Let (D,W) be a default theory and I its Konolige's translation. Let S be closed under Cn. Then S is an extension of (D,W) if and only if $E(S)$ is a robust expansion of I.*

A corollary immediately follows.

Corollary 3.7 *Let $I \subseteq \mathcal{L}_K$ be in a normal form and let (D,W) be a default theory such that $tr_K(D,W) = I$. Let $T \subseteq \mathcal{L}_K$ be stable. Then T is a robust expansion of I if and only if $T \cap \mathcal{L}$ is an extension of (D,W).*

The proof of Theorem 3.6 takes up the rest of the section. It is based on the following two lemmas.

Lemma 3.8 *For every closed under consequence theory $S \subseteq \mathcal{L}$, and every default theory (D,W), if $I = tr_K(D,W)$, then $R_n^{D,S}(W) \subseteq B_n^{E(S)}(I)$.*

Proof. We proceed by induction on n. For $n = 0$, $R_0^{D,S}(W) = Cn(W) = Cn(W(I)) = B_0^{E(S)}(I)$ (second equality follows by Remark 3.1). So, assume that $R_n^{D,S}(W) \subseteq B_n^{E(S)}(I)$, and recall that $R_{n+1}^{D,S}(W) = Cn(R_n^{D,S}(W) \cup F)$, where

$$F = \{c(d) : d \in D \wedge p(d) \in R_n^{D,S}(W) \wedge \forall_{\beta\in j(d)} \neg\beta \notin S\}.$$

By the induction hypothesis, $R_n^{D,S}(W) \subseteq B_n^{E(S)}(I)$. Moreover, $B_n^{E(S)}(I) \subseteq B_{n+1}^{E(S)}(I)$. Hence we need to prove that $F \subseteq B_{n+1}^{E(S)}(I)$. So, consider a default $d \in D$ such that $p(d) \in R_n^{D,S}(W)$ and $\neg\beta \notin S$ for every $\beta \in j(d)$. By the induction hypothesis, $p(d) \in B_n^{E(S)}(I)$ and, consequently, $Kp(d) \in KB_n^{E(S)}(I)$. Also, since $S = E(S) \cap \mathcal{L}$, we have $\neg\beta \notin E(S)$ for all $\beta \in j(d)$. Hence, $\neg K\neg\beta \in \neg K\overline{E(S)}$ for all $\beta \in j(d)$. Note that $tr_K(d) = Kp(d) \wedge \neg K\neg\beta_1 \wedge \ldots \wedge \neg K\neg\beta_n \Rightarrow c(d)$. Thus, by modus ponens, $c(d) \in Cn(\{tr_K(d)\} \cup KB_n^{E(S)}(I) \cup \neg K\overline{E(S)})$. In addition, $c(d) \in \mathcal{L}$. Consequently, $c(d) \in B_{n+1}^{E(S)}(I)$. This implies $F \subseteq B_{n+1}^{E(S)}(I)$ and completes the argument for the inductive step and the lemma. □

Lemma 3.9 *Let $S \subseteq \mathcal{L}$ be such that $Cn(S) = S$. Let $I = tr_K(D,W)$, where (D,W) is a default theory, and let $R_\infty^{D,S}(W) \subseteq S$. Then for all $n \in \mathcal{N}$, $B_n^{E(S)}(I) \subseteq R_n^{D,S}(W)$.*

Proof. We proceed by induction on n. For $n = 0$ we have $B_0^{E(S)}(I) = Cn(W(I)) = Cn(W) = R_0^{D,S}(W)$. So, assume that $B_n^{E(S)}(I) \subseteq R_n^{B,S}(W)$ and consider $\gamma \in B_{n+1}^{E(S)}(I)$. Then there exists a default $d = \frac{\alpha:\beta_1\ldots\beta_r}{\omega}$ such that $\gamma \in Cn(F)$, where

$$F = tr_K(d) \cup B_n^{E(S)} \cup KB_n^{E(S)} \cup \neg K\overline{E(S)}.$$

If $\gamma \in B_n^{E(S)}(I)$ then, by the induction hypothesis, $\gamma \in R_n^{D,S}(I) \subseteq R_{n+1}^{D,S}(I)$. So, now assume that $\gamma \notin B_n^{E(S)}(I)$. Since $B_n^{E(S)}(I)$ is closed under consequence, there is a valuation v of $\mathcal{L}$ such that $v(\gamma) = 0$ and $v(\xi) = 1$ for all $\xi \in B_n^{E(S)}(I)$.

Suppose now that $\neg\beta_i \in E(S)$, for some $1 \leq i \leq r$. Extend v to a valuation v_1 of $\mathcal{L}_K$ defining:

- $v_1(K\neg\beta_i) = 1$,

- $v_1(K\phi) = 1$ for all $\phi \in B_n^{E(S)}(I)$, and
- $v_1(K\phi) = 0$ for $\phi \in \overline{E(S)}$.

Such a valuation v_1 exists since $B_n^{E(S)}(I) \subseteq R_n^{D,S}(W) \subseteq R_\infty^{D,S}(W) \subseteq S \subseteq E(S)$. Under valuation v_1, $v_1(\gamma) = 0$ and $v_1(\xi) = 1$ for all $\xi \in F$, a contradiction. (The fact that $v_1(K\neg\beta_i) = 1$ implies that $v(tr_K(d)) = 1$, the rest is straightforward.) Hence $\neg\beta_i \notin E(S)$ and so $\neg\beta_i \notin S$ for all $i \leq r$.

Now, assume that $\alpha \notin B_n^{E(S)}(I)$. Extend v to a valuation v_2 of $\mathcal{L}_K$ by setting:

- $v_2(K\phi) = 1$ for $\phi \in B_n^{E(S)}(I)$, and
- $v_2(K\phi) = 0$ for $\phi \in \overline{E(S)} \cup \{\alpha\}$.

As before, we find that such an extension exists and that $v_2(tr_K(d)) = 1$. Hence, v_2 has value 1 on F and value 0 on γ, a contradiction again.

Thus, $\alpha \in B_n^{E(S)}(I)$. Since $\neg\beta_i \notin S$ for all $i \leq r$, we obtain that $K\alpha \wedge \neg K\neg\beta_1 \wedge \ldots \wedge \neg K\neg\beta_r$ belongs to $Cn(F)$. Consequently, $\omega \in Cn(F)$. In addition, since by the induction hypothesis, $B_n^{E(S)}(I) \subseteq R_n^{D,S}(W)$, we obtain $\alpha \in R_n^{D,S}(W)$. Using again the fact that $\neg\beta_i \notin S$ for all $i \leq r$, we obtain that $\omega \in R_{n+1}^{D,S}(W)$.

Now, we claim that $\gamma \in Cn(B_n^{E(S)}(I) \cup \{\omega\})$. Otherwise, there is a valuation v of $\mathcal{L}$ which has value 1 on $B_n^{E(S)}(I) \cup \{\omega\}$ and 0 on γ. Extend this valuation to a valuation v' of $\mathcal{L}_K$ setting:

- $v'(K\phi) = 1$ for $\phi \in B_n^{E(S)}(I)$, and
- $v'(K\phi) = 0$ for $\phi \in \overline{E(S)}$.

Then $v'(\gamma) = 0$ and $v'(\xi) = 1$ for all $\xi \in Cn(F)$ ($v'(\omega) = 1$ implies that $v'(tr_K(d)) = 1$). This is a contradiction. Consequently, $\gamma \in Cn(B_n^{E(S)}(I) \cup \{\omega\}) \subseteq R_{n+1}^{D,S}(W)$ ($B_n^{E(S)} \subseteq R_n^{D,S}(W) \subseteq R_{n+1}^{D,S}(W)$, and $\omega \in R_{n+1}^{D,S}(W)$). This completes the proof of the lemma. □

Proof of Theorem 3.6. Assume that S is an extension of (D,W). Then $S = \bigcup_{n=1}^{\infty} R_n^{D,S}(W)$ and, by Lemma 3.8, $S \subseteq \bigcup_{n=1}^{\infty} B_n^{E(S)}(I)$ (where $I = tr_K(D,W)$). Next, since S is an extension of (D,W), Lemma 3.9 applies and $\bigcup_{n=1}^{\infty} B_n^{E(S)}(I) \subseteq \bigcup_{n=1}^{\infty} R_n^{D,S}(W) = S$. Hence $S = \bigcup_{n=1}^{\infty} B_n^{E(S)}(I)$. This means that $E(S)$ is a robust expansion of $I = tr_K(D,W)$.

Now, assume that $E(S)$ is a robust expansion of $I = tr_K(D,W)$ and that S closed under consequence. Then $S = E(S) \cap \mathcal{L}$ by Proposition 2.2.1. Hence, by Lemma 3.8, $R_\infty^{(D,S)}(W) = \bigcup_{n=1}^{\infty} R_n^{D,S}(W) \subseteq \bigcup_{n=1}^{\infty} B_n^{E(S)}(I) = S$. Now, Lemma 3.9 applies again and $S = B_\infty^{E(S)}(I) \subseteq R_\infty^{D,S}(W)$. Thus $S = R_\infty^{(D,S)}(W)$, i.e., S is an extension of (D,W). □

4 Relationships between various structures in default logic and autoepistemic logic

In this section we present a broad discussion of interconnections between default and autoepistemic logic. Our starting point is an observation by Konolige that extensions do not form a perfect match for expansions; expansion is a weaker notion than extension. Our main goal in this section is to strengthen the notion of expansion so that it corresponds to the notion of extension, and to weaken the notion of extension so that it corresponds to the notion of expansion. To make it more precise, we formulate the following two problems.

Problem 4.1 *Given a default theory $\Delta = (D,W)$, find a theory $T_\Delta \subseteq \mathcal{L}_K$ and some epistemic "structure$_1$" for T_Δ (related to expansions) such that for $S \subseteq \mathcal{L}$ closed under consequence, S is an extension for (D,W) if and only if $E(S)$ is a "structure$_1$" for T_Δ.*

We gave one solution to Problem 4.1 in Section 3. Its technically complex and too "procedural". The crux of our construction is to simulate faithfully Reiter's operator in autoepistemic realm. In this section we shall give another, more "declarative" and more elegant solution to this problem.

Our second problem is concerned with the idea of weakening the notion of extension.

Problem 4.2 *Given a theory $I \subseteq \mathcal{L}_K$, find a default theory (D_I, W_I) and some "structure$_2$" for (D_I, W_I) (related to extensions) such that if T is stable then T is an expansion for I if and only if $T \cap \mathcal{L}_K$ is a "structure$_2$" for (D_I, W_I).*

Both problems will be solved in this section. Our solution to Problem 4.1 depends on the solution to Problem 4.2. Hence we discuss them in this order. Finally, we discuss interconnections between notions of default and autoepistemic logics based on the principle of parsimony (minimal sets closed under defaults) and stable theories with minimal objective parts. The lack of space does not permit to provide the proofs of the results discussed in this section. These proofs will be included in the full version of the paper.

4.1 Weak extensions and expansions

In Section 2.1 we introduced the notion of weak extension. It is related to the notion of weak proof which, when using defaults as inference rules, allows to apply context not only to the justification part of the default (as is the case with strong proofs that define extensions) but also to the prerequisite part. As each strong proof is a weak proof we have the following result implicitly present already in [Reiter, 1980].

Theorem 4.1.1 *Every extension of (D,W) is a weak extension of (D,W).*

Due to the "symmetric" way context is used in weak proofs, weak extensions turn out to correspond exactly to expansions; note that in the definition of expansion context is also used "symmetrically".

Theorem 4.1.2 *Let (D,W) be a default theory.*
(a) If $S \subseteq \mathcal{L}$ be closed under consequence then S is a weak extension of (D,W) if and only if $E(S)$ is an expansion of $tr_K(D,W)$.
(b) If $T \subseteq \mathcal{L}_K$ is stable then T is an expansion of $tr_K(D,W)$ if and only if $T \cap \mathcal{L}$ is a weak extension of (D,W).

This theorem is not a solution to Problem 4.2 yet. Note that it is concerned with the "converse" to Problem 4.2. The theorem shows that if the interpretation of Konolige is used then weak extensions of a default theory exactly correspond to expansions of its interpretation. In Problem 4.2 we look for an interpretation of an autoepistemic theory as a default theory. There is a natural candidate for such interpretation in the case of autoepistemic theories consisting of ae-defaults (the converse translation to the one of Konolige). Situation is more complicated if an arbitrary autoepistemic theory I is considered. In this case we first use Proposition 2.2.6 to find another theory I^+ consisting of ae-defaults and having the same expansions as I and then apply the converse Konolige translation to theory I^+, to find an appropriate default theory. Let us note that although I^+ is difficult to describe explicitly, it is effectively constructible, algorithms can be found in [Marek and Truszczyński, 1988]. Let then $tr_{dl}(I)$ be the default theory constructed from I in the two steps described above. By Proposition 2.2.6, I^+ has the same expansions as I. Thus, the solution to Problem 4.2 is now an easy consequence of Theorem 4.1.2.

Theorem 4.1.3 *For every autoepistemic theory $I \subseteq \mathcal{L}_K$, and for every stable set $T \subseteq \mathcal{L}_K$, T is an expansion of I if and only if $T \cap \mathcal{L}$ is a weak extension of $tr_{dl}(I)$.*

Hence, when we speak about *weak* extensions and expansions, default and autoepistemic logic are equivalent. Let us look why the fact that the defaults are used as inference rules has no effect here. The reason is, that the context is used both in the prerequisite and the justification parts. Let us consider the theory $\{Kp \Rightarrow p, \neg Kp \Rightarrow p\}$, again. The corresponding default theory is $(\{\frac{p:}{p}, \frac{:\neg p}{p}\}, \emptyset)$. Since the same context is used to both parts, one of the defaults is always applicable and allows to derive p.

Close analysis of the concepts of proof related to the extensions and weak extensions shows that they differ in that we do not allow to use the context in the case of positive part of the default in the case of extension and we do allow it in case of weak extension. Let us define a *prerequisite-free* default as one which has no prerequisite. Similarly, a theory in which all defaults are prerequisite–free is called *prerequisite-free*. We have then the following theorem.

Theorem 4.1.4 *For the prerequisite-free default theories notions of weak extension and extension coincide. Consequently, if I is a theory consisting of ae–defaults in $\mathcal{L}_K$ such that for every formula $A \Rightarrow \omega$ in I, A has no positive modal atom, then every expansion of I is a robust expansion. In particular, objective parts of expansions of I form an antichain.* [1]

More results on connection of extensions and weak extensions can be found in [Zhang and Marek, 1989].

4.2 Declarative form of equivalence of extensions and robust expansions

In this part of the paper we again look at the equivalence problem for extensions and (some) expansions. The solution to Problem 4.1 we present in this section differs significantly from the solution given in Section 3. In order to find a perfect match for extensions we define a class of expansions not by simulating the way Reiter's operator works and using each autoepistemic formula separately (i.e., by means of operator B^T), but rather by means of operator A^T. However, to eliminate unwanted interactions of formulas, we apply A^T not to the whole theory but only to its appropriately chosen part. As in Section 3 we restrict here to autoepistemic theories consisting of ae-defaults only. Let I be such a theory and let $T \subseteq \mathcal{L}_K$. Define $GD(I,T)$ to consist of those formulas $K\alpha \wedge \neg K\psi_1 \ldots \neg K\psi_k \Rightarrow \omega$ of I for which $\alpha \in T$ and $\psi_i \notin T$ for $1 \leq i \leq k$. This is the modification of I we alluded to above. We say that T is a *strong iterative expansion* of a set of ae–defaults I if and only if T is an expansion of I and $A^T(GD(I,T)) = T$. (It can be proved that this is equivalent to $A^T(GD(I,T)) = T$ and $I \subseteq T$.) Set $GD(I,T)$ is a counterpart to the set of generating defaults $GD(\Delta,S)$ introduced in Section 2.1. It turns out that generating defaults play important role in the solution of Problem 4.1. We have the following two auxiliary facts explicitly involving the notion of generating defaults.

Theorem 4.2.1 *Let $\Delta = (D,W)$, $I = tr_K(D,W)$ and let T be an expansion of I. Let $S = T \cap \mathcal{L}$ and assume that $D = GD(\Delta,S)$. Then S is an extension of Δ if and only if $T = A^T(I)$.*

Theorem 4.2.2 *Let $\Delta = (D,W)$ be a default theory and S its weak extension. Let $D' = GD(\Delta,S)$. Theory S is an extension for $\Delta' = (D',W)$ if and only if it is an extension for Δ.*

These two results in combination with Theorems 4.1.2 and 3.2 yield the solution to Problem 4.1.

[1] Halina Przymusińska informed us that she and Michael Gelfond independently proved the last part of Theorem 4.1.4.

Theorem 4.2.3 *Let (D,W) be a default theory.*
(a) *If $S \subseteq \mathcal{L}$ is closed under consequence then S is an extension of (D,W) if and only if $E(S)$ is a strong iterative expansion of $tr_K(D,W)$.*
(b) *If $T \subseteq \mathcal{L}_K$ is stable then T is a strong iterative expansion of $tr_K(D,W)$ if and only if $T \cap \mathcal{L}$ is an extension of (D,W).*

In the special case of autoepistemic theories consisting of ae-defaults only, it is easy to describe the converse interpretation that will assign a default theory to a theory consisting of ae-defaults so that robust expansions correspond to extensions (Konolige's converse translation is adequate for that purpose). Hence, for that particular case, an analogous result to Theorem 4.1.3 holds (see Corollary 3.7). The general problem of interpretation of autoepistemic theories (consisting of arbitrary formulas) as default theories so that extensions correspond to some sort of expansions is much more difficult. It is unclear how this new class of expansions should be defined. Note that robust expansions are defined only for theories consisting of ae-defaults. This definition cannot be extended to the general case by a construction similar to the one used in Proposition 2.2.1 because the notion of robust expansion is "representation-sensitive", i.e., logically equivalent theories of ae-defaults may have different robust expansions (see Section 3).

A careful inspection of the argument shows that a collection of defaults slightly bigger than $GD(\Delta,S)$ (or $GD(I,E(S))$) does not lead to harmful interplay of translations of defaults. This collection was discovered by Konolige in [1988b]. The class proposed by Konolige consists of translations of defaults d such that conclusion of d belongs to T, and $\neg\beta \notin T$ for every $\beta \in j(d)$.

4.3 Structures associated with the parsimony principle

As pointed by various authors [Etherington, 1988, Hanks and McDermott, 1986, Reiter, 1980] while thinking about structures associated with default theories an important feature of agent's belief set is its minimality among sets closed under defaults. It is reasonable to consider those minimal sets on their own. Formally, we say that $S \in \mathcal{L}_K$ is a *minimal set* for a default theory (D,W) if $W \subseteq S$, S is closed under consequence, and under all defaults from D, i.e.,

$$\forall_{d \in D}(p(d) \in S \wedge \forall_{\beta \in j(d)} \neg\beta \notin S \Rightarrow c(d) \in S).$$

In addition, we require that no proper subset of S has these properties.

The notion of a minimal set is different from that of extension. Although not explicitly stated the following fact is implicit in [Reiter, 1980].

Proposition 4.3.1 *Every extension of (D,W) is a minimal set for (D,W).*

Example 4.3.2 *Let $W = \emptyset$, $D = \{\frac{:p}{\neg p}\}$. Theory (D,W) has no extension, but $Cn(\neg p)$ is a (unique) minimal set for (D,W).*

The existence of minimal sets is an easy consequence of Zorn's lemma.

Theorem 4.3.3 *For every (D,W) there exists a minimal set.*

A natural candidate for the autoepistemic counterpart to minimal sets are stable sets minimal with respect to the relation defined as follows: if T_1, T_2 are stable theories then

$$T_1 \sqsubseteq T_2 \text{ iff } T_1 \cap \mathcal{L} \subseteq T_2 \cap \mathcal{L}.$$

The ordering $\sqsubseteq$ is non-trivial and, it follows from [Marek, 1986] that it is isomorphic to the inclusion ordering among the theories in $\mathcal{L}$ closed under consequence.

We have now the following two theorems establishing exact relationship between minimal sets for default theories and $\sqsubseteq$-minimal stable sets. The first one is analogous to Theorems 4.1.2 and 4.2.3, the second one is analogous to Theorem 4.1.3 and Corollary 3.7.

Theorem 4.3.4 *Let (D,W) be a default theory.*
(a) *If $S \subseteq \mathcal{L}$ is closed under consequence then S is a minimal set for (D,W) if and only if $E(S)$ is a $\sqsubseteq$-minimal stable theory containing $tr_K(D,W)$.*
(b) *If $T \subseteq \mathcal{L}_K$ is stable then T is a $\sqsubseteq$-minimal theory containing $tr_K(D,W)$ if and only if $T \cap \mathcal{L}$ is a minimal set for (D,W).*

Theorem 4.3.5 *For every autoepistemic theory $I \subseteq \mathcal{L}_K$, and for every stable set $T \subseteq \mathcal{L}_K$, T is $\sqsubseteq$-minimal for I if and only if $T \cap \mathcal{L}$ is a minimal set for $tr_{dl}(I)$.*

5 Conclusions

The problem of relationship between structures naturally appearing as possible sets of consequences by an agent reasoning with defaults and those related to introspective reasoning has been cleared up in this paper. *As long as we deal with default theories and their translations (i.e. ae-defaults)* the picture is complete: extensions correspond to robust (i.e. strongly iterative) expansions, weak extensions to expansions and minimal sets to $\sqsubseteq$-minimal stable sets.

The situation becomes more complicated if we look at the picture from the autoepistemic side, and in particular if we consider theories I with formulas with nested occurrences of the operator K. In that case the situation is this: The exact match is obtained for expansions and $\sqsubseteq$ -minimal stable sets. The problem with robust expansions is that there is more than one representation of a given theory I (with nested K) in form of ae-defaults. Different such representations, although logically equivalent, may – and in fact will, have different representations as default theories, in

particular with different extensions. This dependence on syntactic representation (absent in case of arbitrary expansions and $\sqsubseteq$–minimal stable sets) makes the situation complex.

It is, however, worth mentioning that these results work for the Konolige's translation only. It is easy to establish different translations of default theories into the autoepistemic logic and the results may be quite different. The Konolige's translation seems to capture, however, the basic intuition that once a formula is proved, it should be accepted as known. Our results, and those of Konolige as well, show that there is a potentially rich structure theory for expansions. We hope that further studies allow for subtler analysis of the way a fully retrospective agent thinks.

Another aspect of results established in this paper is that the arguments of Moore [1985] indicating the difference between default and introspective reasonings have to be taken with a grain of salt. Although technically different, both modes of reasoning are bi-interpretable; in fact we see that respective reasonings can be simulated in the other theory faithfully. As concerns the primary modes of reasoning in autoepistemic logic and default logic by means of extensions and expansions respectively, the difference seems to be that whereas the context is used symmetrically in autoepistemic reasonings of Moore (both on positive and negative side of context–dependent reasoning), it is used on the negative side only in the default reasonings of Reiter.

Our representation of extensions of default theories by means of fixed points of (parametrized) monotone operators provides a clear procedure which (at least in the propositional case) makes the problem of computing extensions decidable. It is also obvious, that search for extensions can be performed without complete search through all theories.

Finally, let us notice one more important aspect of the above research. The Konolige's interpretation and our result on the perfect match between extensions and robust expansions provides a semantics for default logic. This is a two–step construction: Firstly we translate default theory to $\mathcal{L}_K$ via Konolige's translation, find expansions of the translation (i.e. correct modal indices for list semantics of Moore) and then leave only those which are robust. Although the construction is certainly involved, it provides a starting point for the investigations of semantically defined consequence operators for defaults (i.e. allows to define when default theory entails *a default*). It is certainly possible to provide other semantics for default logic. The question of finding one that is both natural and, possibly, complete remains still open.

6 Acknowledgements

We acknowledge helpful suggestions of and conversations with Francisco Corella, David Etherington, Kurt Konolige, Halina Przymusińska, Raymond Reiter and Aidong Zhang.

References

[Etherington, 1988] D. W. Etherington. *Reasoning with Incomplete Information.* Pitman, London, 1988.

[Hanks and McDermott, 1986] S. Hanks, D. McDermott. Nonmonotonic Logic and Temporal Projection. *Artificial Intelligence* 33:379–412, 1986.

[Konolige, 1988a] K. Konolige. On the Relation between Default and Autoepistemic Logic. *Artificial Intelligence* 35:343–382, 1988.

[Konolige, 1988b] K. Konolige. *A private communication*, 1988.

[Levesque, 1987] H.J. Levesque. All I know: An Abridged Report. In *Proceedings of AAAI Conference*, pages 426–431, 1987

[Marek, 1986] W. Marek. Stable Theories in Autoepistemic Logic. To appear in *Fundamenta Informaticae.*

[Marek and Truszczyński, 1988] W. Marek and M. Truszczyński. Autoepistemic Logic, submitted. (Available as Technical Report 115-88, Department of Computer Science, University of Kentucky, Lexington, KY 40506-0027, 1988)

[Marek and Truszczyński, 1989] W. Marek and M. Truszczyński. Stable semantics for logic programs and default theories, submitted.

[Moore, 1985] R.C. Moore. Semantical Considerations on Non-Monotonic Logic. *Artificial Intelligence*, 25:75–94, 1985.

[Moore, 1984] R.C. Moore. Possible-world semantics for Autoepistemic Logic, Unpublished Note, 1984.

[Reiter, 1980] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Reiter, 1987] R. Reiter. Nonmonotonic Reasoning. *Ann. Rev. Comput. Sci.*, 2:147–186, 1987.

[Stalnaker, 1980] R.C. Stalnaker. A note on nonmonotonic modal logic. Unpublished Manuscript, 1980.

[Zhang and Marek, 1989] A. Zhang and W. Marek. On the classification and existence of extensions in default logic. Technical Report 137-89, Department of Computer Science, University of Kentucky, Lexington, KY 40506-0027, 1989.

Taxonomic Syntax for First Order Inference

David McAllester
Bob Givan
Tanveer Fatima
MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge Mass. 02139

Abstract

Most knowledge representation languages are based on classes and taxonomic relationships between classes. Taxonomic hierarchies without defaults or exceptions are semantically equivalent to a collection of formulas in first order predicate calculus. Although designers of knowledge representation languages often express an intuitive feeling that there must be some advantage to representing facts as taxonomic relationships rather than first order formulas, there are few, if any, technical results supporting this intuition. We attempt to remedy this situation by presenting a taxonomic syntax for first order predicate calculus and a series of theorems that support the claim that taxonomic syntax is superior to classical syntax.

1 Introduction

Most knowledge representation languages are based on classes and taxonomic relationships between classes [Bobrow and Winograd, 1977], [Fahlman, 1979], [Brachman, 1983], [Brachman *et al.*, 1983]. Taxonomic hierarchies without defaults or exceptions are semantically equivalent to a collection of formulas in first order predicate calculus. Designers of knowledge representation languages have argued that there are computational advantages to representing facts as taxonomic relationships rather than first order formulas. However, these arguments are usually non-technical, appealing to the reader's intuition and common sense rather than technical analysis.

We define a taxonomic syntax for first order predicate calculus. In this syntax terms are generalized to the notion of a class expression. Each class expression denotes a subset of the first order domain and all atomic formulas are simple statements about class expressions. We show that the quantifier-free taxonomic literals, i.e. atomic formulas or their negations[1] are more expressive than literals of classical first order logic. For example, there exists a set of two quantifier-free taxonomic literals that is satisfiable but is not satisfied by any finite first order structure — any satisfiable set of literals in the classical predicate calculus with equality can be satisfied by some finite structure. In spite of the increased expressive power of taxonomic literals, we show that the satisfiability of any set of quantifier-free taxonomic literals is polynomial time decidable.

The two basic observations about taxonomic syntax—that quantifier-free taxonomic literals are more expressive than classical literals, and that the satisfiability of a set of quantifier-free taxonomic literals is polynomial time decidable—suggest that taxonomic syntax is more powerful, in some way, than classical syntax. However, these observations do not provide any clear way of taking advantage of taxonomic syntax in general theorem proving. To show the value of taxonomic syntax in general theorem proving, we define a "high-level" proof system based on a strengthened version of the decision procedure for the decidability of a set of quantifier-free taxonomic literals. The strengthened decision procedure provides a technical notion of an "obvious" step in a mathematical proof; a high-level proof is a sequence of steps where each step obviously follows from previous steps.

There is a continuum between theorem verification and theorem proving. No modern theorem proving system can automatically find proofs of theorems as hard as the prime factorization theorem in number theory. A man-machine interactive system, however, can be used to verify such theorems [Bledsoe, 1977], [Boyer and Moore, 1979], [Constable *et al.*, 1985], [Ketonen, 1984] [McAllester, 1989]. Without powerful theorem proving mechanisms the amount of user-provided detail required is so large that non-trivial verifications are impractical. As the requirement for

[1] In taxonomic syntax it is possible for atomic formulas to contain quantifiers; the decidability result only applies to sets of quantifier-free taxonomic literals.

user-provided detail decreases, a verification system can make a continuous transformation from being a proof verifier to a proof finder. Thus the classification of systems into verifiers and provers is somewhat arbitrary. A high-level proof system combines the notion of a user-specified proof with the notion of a sophisticated theorem-proving procedure that determines the correctness of individual proof steps. The decision procedure for proof-step correctness should always terminate quickly.

Many of the features of the high-level proof system introduced here, such as focus objects and rules of obviousness, are independent of taxonomic syntax. These features of high-level proof systems were introduced by McAllester in the Ontic theorem verification system, [McAllester, 1989], and found to be effective in a machine verification of a proof of the Stone representation theorem for Boolean lattices from the axioms of Zermelo-Fraenkel set theory. The high-level proof system introduced by McAllester is not based on taxonomic syntax. In this paper we argue in favor of taxonomic syntax by comparing the length of high-level proofs in a system based on classical syntax with the length of proofs in an analogous system based on taxonomic syntax. We show that any proof in classical syntax can be translated into a proof of the same length in taxonomic syntax. Furthermore, we conjecture that the converse is not true, i.e., we conjecture that there exist proofs in taxonomic syntax such that all classical syntax proofs of the same result are much longer.

2 Taxonomic Syntax for First Order Logic

Our taxonomic syntax for first order logic is organized around classes and taxonomic formulas. Consider a model of first order logic. Each class expression of taxonomic syntax denotes a subset of the domain, or universe of discourse, of the first order model. The class expressions include ordinary first order terms as a special case. Under the semantics of taxonomic expressions, terms are class expressions that denote singleton sets. But there are many class expressions that are not terms in the ordinary sense. For example, a a predicate symbol P of one argument is a class expression denoting the set of all objects in the first order domain that satisfy the predicate P. If $s_1 \ldots s_k$ are class expressions, and f is a function symbol which takes k arguments, then $(f\, s_1 \ldots s_k)$ is also a class expression and denotes the set of all elements which can be written as $(f\, x_1 \ldots x_k)$ where x_i is an element of the set denoted by s_i. Now consider a k-ary predicate symbol R, i.e., a predicate of k arguments. A predicate of k arguments can be viewed as a function which takes $k-1$ arguments and returns a set. More specifically, we can write $(R\, x_1 \ldots x_{k-1})$ to denote the set of all elements y such that $R(x_1 \ldots x_{k-1},\ y)$ is true. If $s_1 \ldots s_{k-1}$ are class expressions then $(R\, s_1 \ldots s_{k-1})$ is also a class expression and denotes the union of all sets of the form $(R\, x_1 \ldots x_{k-1})$ where x_i is an element of s_i. A class expression completely constructed from variables, constants, and function symbols will be called a term. Terms always denote singleton sets. In addition to the class expressions discussed above, taxonomic syntax allows for classes defined with formulas; one can construct a class expression that denotes the set of all objects x that satisfy an arbitrary formula $\Phi(x)$. In order to ensure that taxonomic syntax is expressively equivalent to classical first order logic, a distinguished class expression, *A-Thing*, always denoted the entire domain in any first order interpretation.

The formulas of taxonomic syntax include atomic statements about the taxonomic relationships between class expressions. More specifically, we write (IS s_1 s_2) to say that the class s_1 is a subset of the class s_2. We also write (THERE-EXISTS s) to say that the class s is non-empty and we write (DETERMINED s) to say that there is at most one element of the class s. Finally, we write (INTERSECTS s t) to say that the class s has a non-empty intersection with the class t.

Definition: A *class expression* is either

- a variable,
- a constant symbol,
- a monadic predicate symbol,
- a k-ary function symbol applied to a k class expressions,
- a k-ary predicate symbol applied to $k-1$ class expressions,
- a such-that expression of the form $(s\ x\ \texttt{S.T.}\ \Phi(x))$ where s is a class expression, x is a variable, and $\Phi(x)$ is a taxonomic formula,
- or the distinguished class expression *A-Thing*.

A *taxonomic formula* is either

- an is-formula, (IS s_1 s_2), where s_1 and s_2 are class expressions,
- an existence-formula, (THERE-EXISTS s), where s is a class expression,
- a determined-formula, (DETERMINED s), where s is a class expression,
- an intersection-formula, (INTERSECTS s t), where s and t are class expressions,
- or a Boolean combination of taxonomic formulas.

Formulas of the first three kinds will be called *atomic formulas*. A *literal* is either an atomic formula or the negation of an atomic formula. A formula or class expression is *quantifier-free* if it does not contain any such-that class expressions.

Given a model of first order logic and an interpretation of every variable as an element of the first order domain, each class expression in taxonomic syntax can be unambiguously interpreted as a subset of the first order domain and each formula of taxonomic syntax can be assigned an unambiguous truth value. For example, the formula (IS x *A-Person*) is true just in case the value of the variable x is an element of the set denoted by the class expression *A-Person*. The formula (IS y (*A-Child-of* x)) is true just in case the relation the pair $<x, y>$ is contained in the relation denoted by *A-Child-of*. The formula (IS z (*A-Child-of* (*A-Child-of* x))) is true just in case there exists some member y of the class (*A-Child-of* x) such that z is a member of the class (*A-Child-of* y). The formula (IS x (*Times* 2 *A-Number*)) is true just in case x can be written as the product of 2 and some number, i.e., just in case x is an even number. The such-that class expression (*A-Person* x S.T. (THERE-EXISTS (*A-Child-of* x))) denotes the set of all people who have children.

Our definition of taxonomic formulas does not include classical quantification. All quantification is done with such-that class expressions. For example, the formula (THERE-EXISTS (*A-Person* x S.T. $\Phi(x)$)) is true just in case there exists some element x of the class *Person* such that $\Phi(x)$ is true. Universal quantification can be defined in terms of existential quantification and negation. Alternatively, one can express universal quantification directly with taxonomic atomic formulas. For example, the formula (IS *A-Person* (*A-Person* x S.T. $\Phi(x)$)) is true if and only if $\Phi(x)$ is true for every member x of the set denoted by *A-Person*.

3 Satisfiability of Quantifier-Free Taxonomic Literals

For class expressions constructed purely from functions and constants, i.e., for classical terms, the IS relation is semantically identical to equality. This implies that every literal in classical first order logic with equality is semantically equivalent to some quantifier-free taxonomic literal. However, most non-trivial quantifier-free taxonomic literals are not equivalent to any classical literal. For example, let P be a monadic predicate symbol and let f be a monadic function symbol. The pair of literals (IS P $f(P)$) and (NOT (IS $f(P)$ P)) is satisfiable. For example, P can be interpreted as the non-negative integers and f as the function that subtracts one from its argument. In this case $f(P)$ denotes the set containing the non-negative integers plus negative one. One can show, however, that this pair of literals can not be satisfied by any *finite* first order structure. Every satisfiable set of literals in classical first order logic with equality can be satisfied by some finite structure.

Since quantifier-free taxonomic literals are more expressive than classical literals, it is not immediately clear whether or not one can efficiently determine the satisfiability of a set of quantifier-free taxonomic literals.

Taxonomic Quantifier-Free Decidability Theorem: The satisfiability of a set of quantifier-free taxonomic literals is polynomial-time decidable.

There is a well known corresponding theorem for classical first order logic; the satisfiability of a set of literals in first order logic with equality is polynomial time decidable. The classical decision procedure is based on the congruence closure algorithm [Kozen, 1977], [Downey *et al.*, 1980], [Nelson and Oppen, 1980]. Unfortunately, the taxonomic decision procedure is significantly more complex than the classical procedure based on congruence closure. To appreciate the complexity of the taxonomic satisfiability problem, consider the literals (IS $f(P)$ a), (IS $f(Q)$ b) and (NOT (IS a b)) where P and Q are monadic predicates, f is a monadic function and a and b are constant symbols. These literals imply that the classes P and Q must be disjoint: if c, say, was in both P and Q, then $f(c)$ must equal both a and b, contradicting the third literal. Now suppose we add the literals (IS c P), (IS $g(c)$ Q), (IS $g^6(P)$ P) and (IS $g^7(Q)$ Q) where c is a constant symbol, g is a monadic function symbol, and $g^n(s)$ abbreviates $g(g(\cdots g(s)))$ with n applications of g. All of these literals taken together are unsatisfiable. To see this it suffices to observe that, under any interpretation, $g^{36}(c)$ must be a member of both P and Q.

Any set of quantifier-free taxonomic literals can be efficiently translated into an equisatisfiable set of quantifier-free literals that does not contain existence, determined, or intersection-formulas. More specifically, both positive and negative literals involving existence, determined, and intersection-formulas can be replaced by literals involving is-formulas and new constant and function symbols. For example, the literal (NOT (INTERSECTS P Q)) can be translated into (IS $f(P)$ a), (IS $f(Q)$ b) and (NOT (IS a b)). Thus, without loss of generality, one can assume that every literal involves an is-formula. It turns out that this

apparent simplification, i.e., the elimination of existence, determined, and intersection-formulas, is not a simplification at all. Our decision procedure relies on existence, determined, and intersection formulas. The decision procedure is based on the following rules of inference for taxonomic literals and intersection formulas. In the following rules s, r, and t range over class expressions, c ranges over constant symbols, f ranges over function symbols, and R ranges over both function and predicate symbols.

(1) $$\frac{(\texttt{IS}\ s_1\ t_1), \cdots (\texttt{IS}\ s_n\ t_n)}{(\texttt{IS}\ f(s_1, \ldots s_n)\ f(t_1, \ldots t_n))}$$

(2) $$\frac{(\texttt{IS}\ r\ s),\ (\texttt{IS}\ s\ t)}{(\texttt{IS}\ r\ t)}$$

(3) $$(\texttt{IS}\ t\ t)$$

(4) $$(\texttt{THERE-EXISTS}\ c)$$

(5) $$(\texttt{DETERMINED}\ c)$$

(6) $$\frac{(\texttt{THERE-EXISTS}\ s_1), \cdots (\texttt{THERE-EXISTS}\ s_n)}{(\texttt{THERE-EXISTS}\ f(s1, \cdots s_n))}$$

(7) $$\frac{(\texttt{DETERMINED}\ s_1), \cdots (\texttt{DETERMINED}\ s_n)}{(\texttt{DETERMINED}\ f(s1, \cdots s_n))}$$

(8) $$\frac{(\texttt{NOT}\ (\texttt{DETERMINED}\ t))}{(\texttt{THERE-EXISTS}\ t)}$$

(9) $$\frac{(\texttt{THERE-EXISTS}\ R(s_1, \ldots s_n))}{(\texttt{THERE-EXISTS}\ s_i)}$$

(10) $$\frac{(\texttt{THERE-EXISTS}\ r),\ (\texttt{IS}\ r\ t)}{(\texttt{THERE-EXISTS}\ t)}$$

(11) $$\frac{(\texttt{DETERMINED}\ t),\ (\texttt{IS}\ r\ t)}{(\texttt{DETERMINED}\ r)}$$

(12) $$\frac{(\texttt{NOT}\ (\texttt{IS}\ r\ t))}{(\texttt{THERE-EXISTS}\ r)}$$

(13) $$\frac{(\texttt{THERE-EXISTS}\ r),\ (\texttt{IS}\ r\ s),\ (\texttt{IS}\ r\ t)}{(\texttt{INTERSECTS}\ s\ t)}$$

(14) $$\frac{(\texttt{INTERSECTS}\ r\ t),\ (\texttt{IS}\ r\ s)}{(\texttt{INTERSECTS}\ s\ t)}$$

(15) $$\frac{(\texttt{INTERSECTS}\ r_1\ s_1), \cdots (\texttt{INTERSECTS}\ r_n\ s_n)}{(\texttt{INTERSECTS}\ f(r_1, \ldots r_n)\ f(s_1, \ldots s_n))}$$

(16) $$\frac{(\texttt{INTERSECTS}\ r\ s)}{(\texttt{INTERSECTS}\ s\ r)}$$

(17) $$\frac{(\texttt{INTERSECTS}\ r\ s)}{(\texttt{THERE-EXISTS}\ s)}$$

(18) $$\frac{(\texttt{INTERSECTS}\ s\ t),\ (\texttt{DETERMINED}\ s)}{(\texttt{IS}\ s\ t)}$$

If Σ is a set of taxonomic literals the notation $\Sigma \vdash\!\circ\ \Psi$ abbreviates the statement that there exists a derivation of Ψ from Σ using the above rules of inference. The notation $\Sigma \vdash\!\circ\ \mathbf{F}$ abbreviates the statement that there exists some formula Ψ such that $\Sigma \vdash\!\circ\ \Psi$ and $\Sigma \vdash\!\circ\ (\texttt{NOT}\ \Psi)$. It is not clear that one can quickly determine whether or not $\Sigma \vdash\!\circ\ \Psi$, or whether $\Sigma \vdash\!\circ\ \mathbf{F}$. However, one can readily construct a decision procedure for a seemingly more restricted inference relation. More specifically, the notation $\Sigma \vdash \Psi$ abbreviates the statement that Ψ can be derived from Σ using the above rules *such that every class expression appearing in the derivation of* Ψ *also appears as a subexpression of some formula in* Σ. The notation $\Sigma \vdash \mathbf{F}$ abbreviates the statement that there exists a formula Ψ such that $\Sigma \vdash \Psi$ and $\Sigma \vdash (\texttt{NOT}\ \Psi)$. Section 4 gives a cubic procedure for determining if $\Sigma \vdash \mathbf{F}$. Section 5 contains a proof that if Σ is a set of quantifier-free taxonomic literals, and $\Sigma \not\vdash \mathbf{F}$, then Σ is satisfiable. This implies that $\Sigma \vdash \mathbf{F}$ if and only if $\Sigma \vdash\!\circ\ \mathbf{F}$ and thus the restricted relation is not really any weaker than the unrestricted relation.

4 The Decision Procedure

Let Σ be a set of quantifier-free taxonomic literals and let Υ be the set of class expressions that appear as subexpressions of members of Σ. The set Υ of class expressions can be viewed as a semantic network where the elements of Υ are viewed as nodes representing classes. The decision procedure for determining whether $\Sigma \vdash \mathbf{F}$ can be viewed as a label-propagation process on this network. More specifically, it is possible to show that if Ψ is a formula not in Σ, but $\Sigma \vdash \Psi$, then Ψ must be a label formula for Υ as defined below.

> **Definition:** A *label formula* for a set Υ of class expressions is a formula of the form (THERE-EXISTS s), (DETERMINED s), (IS s t), or (INTERSECTS s t) where s and t are members of Υ.

Since some of the label formulas involve two members of Υ, it is perhaps better to view them as arcs between nodes rather than labels on nodes. It is possible to determine whether or not $\Sigma \vdash \mathbf{F}$ by propagating labels on the network Υ. More specifically, one continues to derive new label formulas until no more such derivations can be made. If Υ contains n nodes then there are $O(n^2)$ label formulas. Thus the process of deriving new formulas must terminate. If this propagation process yields some label formula Ψ such that Σ contains (NOT Ψ), then $\Sigma \vdash \mathbf{F}$, otherwise $\Sigma \not\vdash \mathbf{F}$.

To analyze the running time of the label propagation procedure it is necessary to specify the procedure in greater detail. In presenting the details of our decision procedure we assume that all class expressions that are applications of a relation or function symbol involve at most two arguments. Expressions involving more than two arguments can be reformulated in terms of expressions that involve only two arguments and thus there is no loss of generality in restricting applications to two arguments. More specifically, if there is a function f of more than two arguments then one simply introduces a new function symbol g and uniformly replaces every class expression of the form $f(s_1,\ s_2 \ldots s_n)$ with $f(s_1,\ g(s_2 \ldots s_n))$. If the new function g takes more than two arguments the process can be repeated. In the worst case this transformation process leads to a linear increase in the length of expressions.

Our procedure runs on a graph-like data structure where each node represents an expression in Υ. This graph-like data structure can be viewed as a directed acyclic graph (DAG) representation of the class expressions in Υ. Each node in this graph is a data structure containing various kinds of information. The data structure representing a class expression s contains fields that are updated whenever a formula of the form (THERE-EXISTS s) or (DETERMINED s) is derived. The data structure representing s also contains a list of all the nodes t such that the formula (IS s t) has been derived, as well as a list of all nodes w such that (IS w s) has been derived, and a list of all nodes u such that (INTERSECTS s u) has been derived. Each time a new label formula is added the procedure must check to see if this addition can be propagated to yield further additional label formulas. There are different propagation procedures corresponding to each kind of label formula. For example there is a propagation procedure that is called when a new formula of the form (IS s t) is derived and a different procedure that is called when a new formula of the form (THERE-EXISTS s) is derived.

Each inference rule is implemented by pieces of propagation procedures. Since there is no way of knowing which antecedent will be derived last, each antecedent of a given rule corresponds to a piece of one of the propagation procedures. For example, consider the first rule of the previous section, the monotonicity rule. For applications involving two arguments, the rule says that if one can derive (IS s t) and (IS u w), then one can derive (IS $R(s,u)$ $R(t,w)$). Each of the two antecedents of this rule corresponds to a piece of the procedure for propagating new is-formulas. Consider the first antecedent, (IS s t). When a new formula (IS s t) is derived a certain piece of the procedure for propagating is-formulas finds all expressions in Υ of the form $R(s,u)$. Expressions of the form $R(s,u)$ are stored on a list in the data structure representing s. For each previously derived formula of the form (IS u w), a hash table lookup is used to see if the expression $R(t,w)$ is in Υ. If so, the formula (IS $R(s,u)$ $R(t,w)$) is derived and, provided that this formula has not been previously derived, the is-formula propagation procedure is called recursively on the new formula. Since there is no way of knowing which antecedent of the rule will be derived last, there is also a piece of the procedure for propagating is-formulas that corresponds to the second antecedent. When a new is-formula (IS u w) is derived, this piece finds all expressions in Υ of the form $R(s,u)$ and then for each previously derived formula (IS s t) looks for the expression $R(t,w)$ in a hash table. This may lead to the recursive addition of another is-formula. Each of the other rules can also be implemented with pieces of propagation procedures; one piece for each antecedent of the rule. Rule 10, for example, can be implemented as a piece of the procedure for propagating existence formulas and a piece of the procedure for propagating is-formulas. Rule 15 is analogous to monotonicity rule and is implemented by pieces of the procedure for propagating is-formulas. The propagation procedures are recursive and no queue of outstanding inferences is required.

The total running time of the propagation process is equal to sum over all rules of the time spent executing the pieces of the propagation procedures that correspond to that rule. For example, consider the monotonicity rule as discussed above. Assuming that hash table lookups take constant time, the time spent executing the monotonicity pieces of the is-formula propagation procedure is bounded by some constant times the total number of hash table lookups performed by these pieces. It is possible to show that for each term $R(s,u)$ in Υ, and each pair of derived is-formulas of the form (IS s t) and (IS u w), there is exactly one hash table lookup performed by the monotonicity pieces of the is-formula propagation procedure; at the point where both is-formulas are derived the expression $R(t,w)$ will be looked up in the hash table. For a fixed expression $R(s,u)$ in Υ, the propagation process can derive at most n^2 pairs of is-formulas of the form

(IS s t) and (IS u w). Therefore, there are at most n^3 hash table lookups performed in the monotonicity pieces of the is-formula propagation procedure.

Assuming that no application expression has more than two arguments, each rule can be implemented so that at most $O(n^3)$ time is spent in the pieces of the propagation procedures that correspond to that rule (where n is the number of class expressions in Υ). Thus, if applications involve at most two arguments, the total time spent in the propagation process is at most $O(n^3)$.

5 Correctness of the Decision Procedure

Suppose that Σ is a set of quantifier-free taxonomic literals. This section summarizes a proof that if $\Sigma \not\vdash \mathbf{F}$ then Σ is satisfiable and thus the procedure of the previous section can determine the satisfiability of Σ. The proof is based on a method for constructing a model of Σ from the set of label formulas Ψ such that $\Sigma \vdash \Psi$. As pointed out earlier, it is possible that Σ is satisfiable and yet there are no finite models of Σ. Thus, the method of constructing a model of Σ must be capable of yielding infinite models. However, the structure of the model is somehow completely characterized by the finite set of formulas Ψ such that $\Sigma \vdash \Psi$.

Let Υ be the set of class expressions that appear as subexpressions of formulas in Σ. The domain elements in any interpretation of Σ can be classified into types depending on their relationships with the class expressions in Υ. More specifically, if d is a domain element of a model of Σ, then the Υ-type of d is defined to be the set of class expressions s in Υ such that d is contained in the set denoted by s. If we view the class expressions in Υ as predicates, then the Υ-type of d is the set of class expressions that are true of d. More generally, an Υ-type is defined to be any subset of the class expressions in Υ. If there are n class expressions in Υ, then there are 2^n different Υ-types. We say that an Υ-type τ is *inhabited* in a particular model of Σ if there exists some domain element d of that model whose Υ-type is τ. Of course, there can be models in which many of the Υ-types are not inhabited.

The model we construct will have have the property that existence formulas and intersection formulas that are not derivable by label propagation will be false in the model. This condition places constraints on the Υ-types that can be inhabited in our model. The types consistent with these constraints are said to be Σ-inhabitable. More specifically, a Σ-inhabitable Υ-type is an Υ-type τ such that $\Sigma \vdash$ (THERE-EXISTS s) for every s in τ, if s is in τ and $\Sigma \vdash$ (IS s w) then w is in τ, and for all s and w in τ, $\Sigma \vdash$ (INTERSECTS s w). If s is a class expression such that $\Sigma \vdash$ (THERE-EXISTS s), then s^* is defined to be the Υ-type consisting of all class expressions w such that $\Sigma \vdash$ (IS s w). If s is a class expression (possibly outside of Υ) such that $\Sigma \not\vdash$ (THERE-EXISTS s) then s^* is defined to be the empty Υ-type. One can show that for any class expression s, s^* is a Σ-inhabitable Υ-type. If $\Sigma \vdash$ (THERE-EXISTS s) then s^* is the least Σ-inhabitable Υ-type that contains s.

It is tempting to define the semantic domain of the desired model of Σ to be the set of Σ-inhabitable types. Unfortunately, this does not allow for infinite domains and Σ may not admit finite models. The need for infinite domains arises from the need to include "predecessors". If the type τ contains a class expression of the form $f(s)$ where f is a function symbol, then for any domain element d that inhabits the type τ there must be some predecessor domain element d' that is an element of the class denoted by s and such that $f(d')$ equals d. If $\Sigma \vdash$ (IS s $f(s)$) then the need to include a predecessor for each element of s may force an infinite domain.

These problems can be solved by taking the domain elements be pairs of the form $<\tau, \alpha>$ where τ is a Σ-inhabitable Υ-type and α is an expression that specifies the role played by the domain element. More specifically, the domain D is inductively defined as follows. Every Σ-inhabitable type must have at least one inhabitant in the model. Thus for, every Σ-inhabitable type τ, D contains the pair $<\tau, 0>$. If a type τ contains a class s such that $\Sigma \vdash$ (DETERMINED s), then τ can have at most one inhabitant; such types will be called term types. If τ is Σ-inhabitable but is not a term type then we require that D contain at least two inhabitants of τ; we specify that D contains the pair $<\tau, 1>$. Finally, if D contains the pair $<\tau, \alpha>$, and τ contains a class expression of the form $f(s_1, \ldots s_n)$, where some s_i^* is not a term type, then D contains the "predecessor" pair $<s_i^*, f(s_1, \ldots s_n) \mapsto <\tau, \alpha>>$ where s_i is the first class expression among $s_1, \ldots s_n$ such that s_i^* is not a term type.

To complete the specification of the model of Σ we must give the interpretation of constant, function, and predicate symbols. A constant c is interpreted to be the pair $<c^*, 0>$. A monadic predicate symbol P is interpreted to be the set of all pairs $<\tau, \alpha>$ where the type τ contains the symbol P. A $k-ary$ predicate symbol R is defined to be the set of tuples $<<s_1^*, 0>, \ldots <s_{k-1}^*, 0>, <\tau, \alpha>>$ such that τ contains the class expression $R(s_1, \ldots s_{k-1})$. Finally, consider applying the function denoted by the symbol f to the arguments $<<\tau_1, \alpha_1>, \ldots <\tau_k, \alpha_k>>$. Suppose there exists some argument $<\tau_j, \alpha_j>$ such that τ_j is not a term type and let $<\tau_i, \alpha_i>$ be the first such argument. Now suppose the expression α_i is a mapping

$f(s_1, \ldots s_n) \mapsto <\sigma, \beta>$ and that s_j^* equals τ_j for each s_j. In this case the value of f on this tuple of arguments is defined to be the pair $<\sigma, \beta>$. If any one of these conditions is not met, then the value of f on this tuple of arguments equals $<\sigma, 0>$ where σ is the union of all types of the form $f(s_1, \ldots s_k)^*$ where each s_j is a member of the type τ_j. The rules of obviousness for intersection-formulas ensure that σ is a Σ-inhabitable Υ-type.

Given the rules of inference listed in section 3 it is possible to prove that under this semantic interpretation the Υ-type of a pair $<\tau, \alpha>$ is, in fact, the type τ. In other words, for any class expression s in Υ, the set denoted by s under the above interpretation equals the set of pairs $<\tau, \alpha>$ such that τ contains s. This fact, and the definition of a Σ-inhabitable Υ-type, can be used to show that the specified interpretation is indeed a model of Σ.

6 Expanded Rules of Obviousness

To compare taxonomic and classical syntax more directly, we define two high-level proof systems: one based on classical syntax and one based on taxonomic syntax. The system based on taxonomic syntax is constructed from a modification of the decision procedure discussed in section 4. This section, and the one that follows, define the high-level proof system based on taxonomic syntax. Given the specification for the taxonomic high-level proof system, the adaptation of that system to classical syntax is presented in section 9.

The first step in defining the high-level proof system is to define a technical notion of an obviously true statement. The obviously true statements are defined by certain rules of obviousness. Each rule of obviousness states that if certain antecedent facts are obvious then a certain conclusion is also obvious. The rules of obviousness contain many, but not all, of the inference rules needed for a complete inference system for first order taxonomic formulas. The rules of obviousness include all of the rules of section 3 together with certain additional rules specified in this section. These additional rules involve a set of variables $\mathcal{F}$ called the *focus set*. We write $\Sigma, \mathcal{F} \vdash\!\circ\ \Psi$ if there exists a derivation of Ψ from the formulas in Σ using the expanded rules of obviousness with focus set $\mathcal{F}$. The notation $\Sigma, \mathcal{F} \vdash\!\circ\ \mathbf{F}$ is analogous to the notation $\Sigma \vdash\!\circ\ \mathbf{F}$ used above.

In taxonomic syntax there are no explicit quantifiers in formulas; all taxonomic formulas are either atomic formulas or Boolean combinations of atomic formulas. Since there are no quantified formulas, no rules of obviousness are needed for quantified formulas. Class expressions, on the other hand, can involve quantifiers. No rules of obviousness have yet been given for such-that class expressions. Intuitively, the rules of obviousness for such-that expressions only allow the such-that quantifier to be instantiated with focus objects. The restriction of the instantiation of quantifiers to focus objects makes it possible to write a procedure for determining obviousness. In the following rules the variable y must be a member of the focus set $\mathcal{F}$.

(19) $$\frac{}{(\texttt{IS}\ (s\ x\ \texttt{S.T.}\ \Phi(x))\ s)}$$

(20) $$\frac{(\texttt{IS}\ y\ s),\ \Phi(y)}{(\texttt{IS}\ y\ (s\ x\ \texttt{S.T.}\ \Phi(x)))}$$

(21) $$\frac{(\texttt{IS}\ y\ (s\ x\ \texttt{S.T.}\ \Phi(x)))}{\Phi(y)}$$

In addition to the above rules for such-that expressions, the expanded rules of obviousness include rules for Boolean connectives. We assume that all Boolean formulas are constructed using the connectives OR and NOT.

(22) $$\frac{(\texttt{OR}\ \Phi\ \Psi),\ (\texttt{NOT}\ \Phi)}{\Psi}$$

$$\frac{(\texttt{OR}\ \Phi\ \Psi),\ (\texttt{NOT}\ \Psi)}{\Phi}$$

(23) $$\frac{(\texttt{NOT}\ (\texttt{OR}\ \Phi\ \Psi))}{(\texttt{NOT}\ \Phi)}$$

$$\frac{(\texttt{NOT}\ (\texttt{OR}\ \Phi\ \Psi))}{(\texttt{NOT}\ \Psi)}$$

(24) $$\frac{(\texttt{NOT}\ (\texttt{NOT}\ \Phi))}{\Phi}$$

It is possible to construct a decision procedure, similar to the one presented in section 3, that determines if $\Sigma, \mathcal{F} \vdash \mathbf{F}$. This decision procedure is based on the idea that the derivations of obvious formulas can be restricted to formulas that only involve certain class expressions. We define a set $\Upsilon(\Sigma, \mathcal{F})$ that contains both formulas and class expressions. More specifically, $\Upsilon(\Sigma, \mathcal{F})$ is defined to be the least set Υ that contains the elements of Σ and $\mathcal{F}$ and such that any subexpression of a member of Υ is also a member of Υ and for every such-that class expression $(s\ x\ \texttt{S.T.}\ \Phi(x))$ in Υ, and every variable y in $\mathcal{F}$, the formula $\Phi(y)$ is also in

Υ. The notation $\Sigma, \mathcal{F} \vdash \Psi$ is defined by analogy with $\Sigma \vdash \Psi$. More specifically, $\Sigma, \mathcal{F} \vdash \Psi$ if there exists a derivation of Ψ using the expanded rules of obviousness such that every class expression appearing in that derivation is a member of $\Upsilon(\Sigma, \mathcal{F})$. The notation $\Sigma, \mathcal{F} \vdash \mathbf{F}$ is defined by analogy with $\Sigma \vdash \mathbf{F}$.

It is possible give a polynomial time procedure for determining whether or not $\Sigma, \mathcal{F} \vdash \mathbf{F}$. As in the previous section, this procedure is based on viewing the set $\Upsilon(\Sigma, \mathcal{F})$ as a set of nodes in a graph analogous to a semantic network. The decision procedure can be viewed as a label-propagation procedure. Since the propagation process is restricted to formulas containing only class expressions in $\Upsilon(\Sigma, \mathcal{F})$, it is possible to show that every formula Ψ such that $\Sigma, \mathcal{F} \vdash \Psi$ is either a formula in Υ, the negation of a formula in Υ or a label formula on the class expression in Υ as defined in section 4.

Unlike the network described in section 4, the network used for the expanded rules of inference contains nodes that represent formulas as well as nodes that represent class expressions. A data structure that represents a formula must be updated whenever that formula is derived using the rules of obviousness, and updated in a different way whenever the negation of the formula is derived. An analysis similar to that given in section 4 shows that the propagation process can be implemented in a way that requires at most $O(n^3)$ time where n is the number of expressions in Υ and assuming that hash table lookups take constant time and that every application class expression involves at most two arguments. As discussed in section 4, there is no loss of generality in assuming that applications involve at most two arguments.

We have not yet ruled out the possibility that $\Sigma, \mathcal{F} \not\vdash \mathbf{F}$ and yet $\Sigma, \mathcal{F} \vdash\!\!\circ\ \mathbf{F}$, i.e. the unbounded inference relation $\vdash\!\!\circ$ may be more powerful than the inference relation $\vdash$ defined by the bounded label-propagation mechanism. It turns out, however, that $\vdash\!\!\circ$ is no more powerful than $\vdash$: the bounded label-propagation procedure is a correct decision procedure for determining whether or not $\Sigma, \mathcal{F} \vdash\!\!\circ\ \mathbf{F}$. The proof that $\vdash\!\!\circ$ is no more powerful than $\vdash$ can not be based on a proof that $\vdash$ is semantically complete: neither $\vdash\!\!\circ$ nor $\vdash$ are complete relative to the intended semantics of Boolean connectives and such-that class expressions. The proof that $\vdash\!\!\circ$ is no more powerful than $\vdash$ is based on a proof technique that we call pseudo-semantic. The basic idea is to define a highly non-standard "semantics" such that $\vdash$ can be shown to be sound and complete relative to that semantics. This pseudo-semantics must also satisfy the condition that all of the inference rules that define $\vdash\!\!\circ$ are sound relative the pseudo-semantics. Since $\vdash\!\!\circ$ is sound, and $\vdash$ is both sound and complete relative to the pseudo-semantics, The relations $\vdash$ and $\vdash\!\!\circ$ must be the same. Unfortunately, space constraints do not allow a detailed discussion of the pseudo-semantic proof that the bounded label-propagation procedure is a correct procedure for determining whether $\Sigma, \mathcal{F} \vdash\!\!\circ\ \mathbf{F}$.

7 A High-Level Proof System

We define the high-level proof system in a way that facilitates formal understanding and technical analysis. A more "user-friendly" specification of an analogous high-level proof system is given in [McAllester, 1989]. In this paper we define a high-level proof to be a series of lines where each line contains a "sequent" of the form $\Sigma \vdash \Phi$ where Σ is a set of formulas and Φ is either a formula or the special token $\mathbf{F}$. The lines of a high-level proof are divided into two kinds: syntactically derived lines and unjustified lines. A syntactically derived line is a line that can be derived from previous lines using one of the following four high-level proof rules. Each high-level proof rule is a form of universal generalization.[2] The need to include rules of universal generalization in the high-level proof system will be discussed in section 8. In the following rules x, and each x_i, must be a variable that does not appear free in any formula in Σ or in any of the class expressions s, t or s_i. In the last rule z must be a variable but there are no restrictions on where z can appear, e.g. z may appear free in Σ or any s_i.

$$\frac{\Sigma \vdash (\texttt{NOT}\ (\texttt{IS}\ x\ s))}{\Sigma \vdash (\texttt{NOT}\ (\texttt{THERE-EXISTS}\ s))}$$

$$\frac{\Sigma \cup \{(\texttt{IS}\ x_1\ s), (\texttt{IS}\ x_2\ s)\} \vdash (\texttt{IS}\ x_1\ x_2)}{\Sigma \vdash \{(\texttt{DETERMINED}\ s)\}}$$

$$\frac{\Sigma \cup \{(\texttt{IS}\ x\ s),\ (\texttt{IS}\ x\ t)\} \vdash \mathbf{F}}{\Sigma \vdash (\texttt{NOT}\ (\texttt{INTERSECTS}\ s\ t))}$$

$$\frac{\Sigma \cup \{(\texttt{IS}\ x\ s)\} \vdash (\texttt{IS}\ x\ t)}{\Sigma \vdash (\texttt{IS}\ s\ t)}$$

$$\frac{\Sigma \cup \{(\texttt{IS}\ x_1\ s_1), \ldots (\texttt{IS}\ x_n\ s_n)\} \vdash (\texttt{NOT}\ (\texttt{IS}\ z\ R(x_1, \ldots x_n)))}{\Sigma \vdash (\texttt{NOT}\ (\texttt{IS}\ z\ R(s_1, \ldots s_n)))}$$

[2] In a user-friendly version of the high-level proof system, each high-level rule of universal generalization appears in its contrapositive form; rather than derive a universal statement from a statement about an arbitrary individual, the user-friendly high-level system allows one to introduce witnesses based on existential statements.

A line of a high-level proof that is not derived from previous lines using one of the high-level generalization rules is called an *unjustified* line. Each unjustified line in a high-level proof must be explicitly associated with a set of variables called the *focus set* of that line. Consider an unjustified line $\Sigma \vdash \Phi$ with associated focus set $\mathcal{F}$. Intuitively, each unjustified line must obviously follow from previous lines in the proof. Let Σ' be Σ plus all formulas previously proven to follow from Σ, i.e., all formulas Ψ such that the proof contains an earlier line of the form $\Gamma \vdash \Psi$ where Γ is a subset of Σ. An unjustified line $\Sigma \vdash \Phi$ with associated focus set $\mathcal{F}$ must follow from previous lines. More specifically, if Φ is the constant $\mathbf{F}$, then we must have $\Sigma', \mathcal{F} \vdash\!\!\circ\ \mathbf{F}$. If Φ is some formula other than $\mathbf{F}$, then we must have $\Sigma' \cup \{(\texttt{NOT}\ \Phi)\}, \mathcal{F} \vdash \mathbf{F}$.

It is important to be able to quickly determine if a series of high-level proof lines is acceptable, i.e. that each unjustified line satisfies the condition specified above. The cost of determining the acceptability of a given unjustified line is extremely sensitive to the size of the focus set $\mathcal{F}$ associated with that line. Fortunately, it seems that in practice the focus sets associated with individual lines can be kept small.[3] More specifically, suppose that there exists some (large) focus set $\mathcal{F}$ such that an unjustified $\Sigma \vdash \Phi$ is acceptable with associated focus set $\mathcal{F}$. Let Σ' be Σ plus all formulas proven to follow from Σ by previous lines of the proof. Let Q be the maximum level of quantifier-nesting that appears in a formula in Σ'. If $\Sigma \vdash \Phi$ is acceptable with a large focus set $\mathcal{F}$, then there exists a sequence of acceptable unjustified high-level proof lines that ends in the line $\Sigma \vdash \Phi$ and where no line involves more than Q focus objects. If we impose a bound on the number of focus objects that can be associated with any single unjustified line in a high-level proof, then the acceptability of a series of high-level proof lines can be determined in polynomial time.

8 High-Level Completeness

It is possible to show that the high-level proof system defined in the previous section is complete for first order taxonomic formulas. More specifically, if a formula Φ semantically follows from a set of formulas Σ, then there exists an acceptable high-level proof that ends with the line $\Sigma \vdash \Phi$. To prove this result one can first observe that there exists a high-level derivation of $\Sigma \vdash \Phi$ if and only if there exists a high-level derivation of $\Sigma \cup \{(\texttt{NOT}\ \Phi)\} \vdash \mathbf{F}$. To prove this it suffices to observe that, given an acceptable derivation of $\Sigma \vdash \Phi$, the line $\Sigma \cup \{(\texttt{NOT}\ \Phi)\} \vdash \mathbf{F}$ can be immediately added as an unjustified line with an empty focus set. Similarly, given any derivation of $\Sigma \cup \{(\texttt{NOT}\ \Phi)\} \vdash \mathbf{F}$, the line $\Sigma \vdash \Phi$ can be acceptably added without justification. To prove the high-level system is complete, we assume that there is no high-level derivation of $\Sigma \vdash \Phi$ and we show that in this case there exists a model of Σ in which Φ is false. If there is no high-level derivation of $\Sigma \vdash \Phi$ then there must not be any high-level derivation of $\Sigma \cup \{(\texttt{NOT}\ \Phi)\} \vdash \mathbf{F}$. To prove that there exists a model of σ in which Φ is false, it now suffices to show that, for any set of formulas Γ, if there is no derivation of $\Gamma \vdash \mathbf{F}$, then there exists some model of Γ.

Suppose that there is no high-level derivation of $\Gamma \vdash \mathbf{F}$. One can construct a model of Γ using techniques analogous to those used in standard proofs of first order completeness. For simplicity we assume that the set of constant, function and predicate symbols in the language is countable and that there is a countably infinite set of variables. In this case one can enumerate all taxonomic formulas in an infinite sequence $\Theta_1, \Theta_2\ \Theta_3 \ldots$.[4] Given that there is no derivation of $\Gamma \vdash \mathbf{F}$, one can then construct an infinite sequence of sets of formulas $\Omega_1, \Omega_2\ \Omega_3 \ldots$ by setting Ω_1 equal to Γ and defining Ω_{j+1} as follows:

1. If there exists a derivation of $\Omega_j \vdash (\texttt{NOT}\ \Theta_j)$ then set Ω_{j+1} equal to Ω_j.
2. If there is no derivation of $\Omega_j \vdash (\texttt{NOT}\ \Theta_j)$, and Θ_j is a formula of the form (THERE-EXISTS s), then let x be some variable that does not appear in s or Ω_j and set Ω_{j+1} to be $\Omega_j \cup \{\Theta_j, (\texttt{IS}\ x\ s)\}$.
3. If there is no derivation of $\Omega_j \vdash (\texttt{NOT}\ \Theta_j)$, and Θ_j is a formula of the form (NOT (DETERMINED s)), then let x and y be variables that do not appear free in s or Ω_j and set Ω_{j+1} to be $\Omega_j \cup \{\Theta_j, (\texttt{IS}\ x\ s), (\texttt{IS}\ y\ s), (\texttt{NOT}\ (\texttt{IS}\ x\ y))\}$.
4. If there is no derivation of $\Omega_j \vdash (\texttt{NOT}\ \Theta_j)$, and Θ_j is a formula of the form (INTERSECTS $s\ t$), then let x be some variable that does not appear free in s, t or Ω_j and set Ω_{j+1} to be $\Omega_j \cup \{\Theta_j, (\texttt{IS}\ x\ s), (\texttt{IS}\ x\ t)\}$.
5. If there is no derivation of $\Omega_j \vdash (\texttt{NOT}\ \Theta_j)$, and Θ_j is a formula of the form (NOT (IS $s\ t$)) where s is not a variable, then let x be some variable that does not appear free in s, t or Ω_j and set Ω_{j+1} to be $\Omega_j \cup \{\Theta_j, (\texttt{IS}\ x\ s), (\texttt{NOT}\ (\texttt{IS}\ x\ t))\}$.
6. If there is no derivation of $\Omega_j \vdash (\texttt{NOT}\ \Theta_j)$ and Θ_j is a formula of the form (IS $x\ R(s_1, \ldots s_n)$)

[3] In the proof of the Stone representation theorem from the axioms of set theory, described in [McAllester, 1989], no unjustified step involved more than ten focus objects. The proof could probably have been done without ever using a focus set of more than four objects.

[4] The completeness proof can be modified to handle uncountable languages, in which case one constructs a transfinite enumeration of formulas.

where x is a variable, then let $y_1, \ldots y_n$ be variables that do not appear in Ω_j or in any of the class expressions s_j, and set Ω_{j+1} equal to $\Omega_j \cup \{\Theta_j, (\texttt{IS}\ x\ R(y_1, \ldots y_j)), (\texttt{IS}\ y_1\ s_1), \ldots (\texttt{IS}\ y_n\ s_n)\}$.

7. If there does not exist a derivation of $\Omega_j \vdash (\texttt{NOT}\ \Theta_j)$, and none of the conditions in 2, 3, 4, or 5 above apply, then set Ω_{j+1} equal to $\Omega_j \cup \{\Theta_j\}$.

Given the high level proof rules introduced in the previous section, one can show that each Ω_j is a finite set of formulas that contains Γ and that there does not exist any derivation of $\Omega_j \vdash \mathbf{F}$. Steps 2, 3, 4, and 5 ensure that, for every form of existential statement that becomes included in the constructed set Ω_{j+1}, there are variables that act as witnesses to that existential statement. For example, if Ω contains the formula (THERE-EXISTS s), then there is some variable x such that Ω contains the formula (IS x s). Steps 2, 3, 4, 5, and 6 in the above specification directly correspond to the five high-level generalization rules presented in section 7. For each of these steps, the proof of the consistency of the newly constructed set Ω_{j+1} relies on the existence of the corresponding high-level generalization rule. Thus, the generalization rules in the high-level proof system are needed because they indirectly allow the introduction of witnesses for existential statements. In a user-friendly high-level proof system the high-level generalization rules can either be used directly or used in the contrapositive form where they allow the introduction of new witnesses to previously proven existential statements.

Now let Ω be the union of all sets Ω_j. It is possible to show that Ω is both consistent and complete. More specifically, for any formula Ψ exactly one of the two formulas Ψ and (NOT Ψ) is contained in Ω. Furthermore, one can show that the set of formulas Ω is closed under all of the rules of obviousness where the rules for such-that expressions are no longer restricted to focus objects.

One can now define a first order structure whose domain consists of equivalence classes of variables. More specifically, for any variable x we define $|x|$ to be the set of variables y such that the formula (IS x y) is a member of Ω. The rules of obviousness for is-formulas ensure that these sets form equivalence classes of variables. We take the domain of the first order structure to be the collection of equivalence classes of the form $|x|$. It is now possible to define an interpretation of the variables, constants, functions, and predicate symbols such that the semantic value of a class expression s equals the set of classes $|x|$ such that the formula (IS x s) is a member of Ω and such that, for every formula Ψ, the semantic interpretation makes Ψ true just in case Ψ is a member of Ω. This provides an interpretation of Γ. Thus one can establish that if there is no derivation of $\Gamma \vdash \mathbf{F}$ then there exists a semantic interpretation of Γ, and similarly, if there is no derivation of $\Sigma \vdash \Phi$, then there exists an interpretation of Σ in which Φ is false.

9 Taxonomic vs. Classical Syntax

To compare taxonomic and classical syntax we consider a high-level proof system analogous to the one defined in section 7 but based on classical rather than taxonomic syntax. A high-level proof in the system based on classical syntax is also a series of lines where each line is "sequent" $\Sigma \vdash \Phi$. Like the taxonomic system, the classical system is based on an obviousness relation $\vdash\!\circ$ and the high-level proof system allows unjustified lines where each unjustified line must be explicitly associated with a set of variables called the focus set for that line. The conditions under which an unjustified line is acceptable are identical in both the taxonomic and classical systems except that the two systems are based on different obviousness relations. Although the obviousness relations underlying the two systems are different, each of the two obviousness relations is defined by a set of inference rules called rules of obviousness.

In the classical system the rules of obviousness presented in section 3 are replaced by the standard rules of inference for equality: reflexivity, symmetry, transitivity, and rules that allow the substitution of equals for equals in terms and atomic formulas. These rules of inference for equality are complete for classical literals: if the rules can not derive a contradiction form a set of first order literals, then the set of literals is satisfiable.

The rules of obviousness that involve Boolean connectives are exactly the same in both the taxonomic and classical systems. In the classical system, we assume that the only quantifier is the classical universal quantifier $\forall$. The three taxonomic rules of obviousness involving such-that class expressions are replaced, in the classical system, by the following single rule of obviousness. In the following rule y must be a variable in $\mathcal{F}$.

$$\frac{\Sigma, \mathcal{F} \vdash\!\circ\ \forall x \Phi(x)}{\Sigma, \mathcal{F} \vdash\!\circ\ \Phi(y)}$$

The five high-level taxonomic generalization rules are replaced, in the classical system, by the following single high-level generalization rule. In the following rule x must be a variable that does not appear free in Σ.

$$\frac{\Sigma \vdash \Phi(x)}{\Sigma \vdash \forall x \Phi(x)}$$

Unlike taxonomic syntax, the classical rules of obviousness involving focus objects makes the relationship between focus objects and previously proven lemmas explicit; the rules of obviousness allow any previously proven universal lemma to be applied to any focus object. In the taxonomic system, a formula of the form $\forall x \Phi(x)$ is represented by the formula (IS *A-Thing* (*A-Thing* x S.T. $\Phi(x)$)). If y is a focus object then the taxonomic rules of obviousness allow the derivation of (IS y *A-Thing*) and given the above is-formula, one can derive (IS y (*A-Thing* x S.T. $\Phi(x)$)). The rules of obviousness for such-that expressions then allow the derivation of $\Phi(y)$. Thus, the above classical rule of universal instantiation for focus objects is subsumed by the taxonomic rules of obviousness. In fact, all of the methods of deriving new lines in the classical high-level proof system are subsumed by methods of deriving new lines in the taxonomic high-level proof system. This claim can be formalized by giving a procedure for translating any proof in the classical high-level system into a corresponding proof in the taxonomic system.

For any classical first order formula Φ, the *taxonomic translation*, $T(\Phi)$ of the formula Φ is defined by structural induction by on Φ. If Φ is an atomic formula of the form $R(s_1, \ldots s_n)$ then $T(\Phi)$ is the taxonomic formula (IS s_n $R(s_1, \ldots s_{n-1})$). $T((\text{OR } \Theta\ \Psi))$ equals (OR $T(\Theta)$ $T(\Psi)$) and $T((\text{NOT } \Psi))$ equals (NOT $T(\Psi)$). If Φ is a universal formula $\forall x \Psi(x)$, then $T(\Phi)$ is the formula (IS *A-Thing* (*A-Thing* x S.T. $T(\Psi(x))$)). For any set Σ of classical first order formulas, $T(\Sigma)$ is the set of taxonomic formulas of the form $T(\Psi)$ for some Ψ in Σ. If P is a high-level proof in the classical high-level proof system, then $T(P)$ is the sequence of lines derived by translating each unjustified line $\Sigma \vdash \Phi$ in P to an unjustified line $T(\Sigma) \vdash T(\Phi)$ leaving the focus set of the line unchanged, and translating each universal generalization line $\Sigma \vdash \forall x \Phi(x)$ to an unjustified line of the form $T(\Sigma) \cup \{(\text{IS } x \text{ } \textit{A-Thing})\} \vdash$ (IS x (*A-Thing* x S.T. $\Phi(x)$)) with focus set $\{x\}$ followed by the generalization line $T(\Sigma) \vdash T(\forall x \Phi(x))$.

> **Taxonomic Domination Theorem:** The taxonomic proof system dominates the classical proof system in the sense that for any acceptable high-level proof P in the classical system, the proof $T(P)$ is acceptable in the taxonomic system.

Intuitively, the proof rules of the taxonomic system include the proof rules of the classical system as a special case. This is not a surprising result and is not difficult to prove. We conjecture, however, that the converse of this theorem does not hold, i.e., the taxonomic high-level proof system is *not* subsumed by the classical high-level proof system.

> **Strict Domination Conjecture:** For any (large) constant k there exists a classical first order formula Φ and a taxonomic proof P of $T(\Phi)$ such that the shortest proof of Φ in the classical high-level proof system has length greater than k times the length of P.

If this conjecture is true, then there would exist a first order statement and a taxonomic proof of that statement such that the shortest classical proof is, say, a hundred times longer than the taxonomic proof.

10 Conclusion

We have defined a taxonomic syntax for first order predicate calculus and have presented several technical results describing computational properties of this syntax. Quantifier-free taxonomic literals are more expressive than literals of classical first order logic and yet there exists a polynomial time decision procedure for determining the satisfiability of a set of quantifier-free taxonomic literals. We have also investigated the value of taxonomic syntax in general theorem proving. We have define high-level proof systems for both taxonomic and classical systems and shown that the taxonomic system subsumes the classical system. Furthermore, we conjecture that the reverse is not true, i.e., that there exist high-level taxonomic proofs such that any classical high-level proof of the same result is much longer.

References

[Bledsoe, 1977] W. W. Bledsoe. Non-resolutuon theorem proving. *Artificial Intelligence*, 9:1–35, 1977.

[Bobrow and Winograd, 1977] D. Bobrow and T. Winograd. An overview of krl, a knowledge representation language. *Cognitive Science*, 1(1):3–46, 1977.

[Boyer and Moore, 1979] Robert S. Boyer and J Struther Moore. *A Compuatational Logic.* ACM Monograph Series. Academic Press, 1979.

[Brachman *et al.*, 1983] R. Brachman, R. Fikes, and H. Levesque. Krypton: A functional approach to knowledge representation. *IEEE Computer*, 16:63–73, 1983.

[Brachman, 1983] R. J. Brachman. What is-a is and isn't: An analysis of taxonomic links in semantic networks. *IEEE Computer*, 16(10):30–36, October 1983.

[Constable *et al.*, 1985]
R.L. Constable, T. B. Knoblock, and J. L. Bates. Writing programs that constuct proofs. *Journal of Automated Reasoning*, pages 285–326, 1985.

[Downey *et al.*, 1980] Peter J. Downey, Ravi Sethi, and Robert E. Tarjan. Variations on the common subexpression problem. *JACM*, 27(4):758–771, October 1980.

[Fahlman, 1979] Scott E. Fahlman. *NETL: A System for Representing Real World Knowledge.* MIT Press, 1979.

[Ketonen, 1984] Jussi Ketonen. Ekl - a mathematically oriented proof checker. In *Proceedings of the Seventh International Conference on Automated Deduction*, pages 65–79, 1984.

[Kozen, 1977] Dexter C. Kozen. *Complexity of Finitely Presented Algebras.* PhD thesis, Cornell University, 1977.

[McAllester, 1989] David A. McAllester. *Ontic: A Knowledge Representation System for Mathematics.* MIT Press, 1989.

[Nelson and Oppen, 1980] Greg Nelson and Derek Oppen. Fast decision procedures based on congruence closure. *JACM*, 27(2):356–364, April 1980.

A Knowledge Level Analysis of Belief Revision

Bernhard Nebel*
IBM Germany, Scientific Center
Stuttgart, West Germany
e-mail: nebel@ds0lilog.bitnet.

Abstract

Revising beliefs is a task any intelligent agent has to perform. For this reason, belief revision has received much interest in Artificial Intelligence. However, there are serious problems when trying to analyze belief revision techniques developed in the field of Artificial Intelligence on the *knowledge level*. The symbolic representation of beliefs seems to be crucial. The *theory of epistemic change* shows that a partial knowledge-level analysis of belief revision is possible, but leaves open the question of how this theory is related to belief revision approaches in Artificial Intelligence. In particular, it remains an open question whether the results achieved in the knowledge-level analysis are valid. Furthermore, the idea of *reason maintenance*, which is considered to be essential in AI, has no counter-part in the theory of epistemic change. Addressing these problems, it is shown how to reconstruct symbol-level belief revision on the knowledge level.

1 Introduction

Any intelligent agent has to account for a changing environment and the fact that its own beliefs might be inaccurate. For this reason, *belief revision* is a task central for any kind of intelligent behavior. For instance, learning [Dietrich, 1986], diagnosis from first principles [Reiter, 1987], and interpretation of counterfactuals [Ginsberg, 1986] are all activities requiring the revision of beliefs.

In Artificial Intelligence, a number of so-called *truth-maintenance systems* [Doyle, 1979, McAllester, 1982, de Kleer, 1986] were developed which support belief revision. However, the question remains how belief revision can be described on an abstract level, independent of how beliefs are represented and manipulated inside a machine. In particular, it is unclear how to describe belief revision on the *knowledge level* as introduced by Newell [1981]. Levesque and Brachman [1986] demanded that every information system should be describable on the knowledge level without any reference to how information is represented or manipulated by the system. However, this seems to be difficult for belief revision. A large number of authors seem to believe that a knowledge-level analysis of belief revision is impossible [Dietrich, 1986, Fagin *et al.*, 1983, Fagin *et al.*, 1986, Ginsberg, 1986]. Considerations of how beliefs are represented on the *symbol level* seem inevitable for belief revision. Reconsidering Newell's original intentions when he introduced the notion of the knowledge level, we note that the main idea was describing the potential for generating actions by knowledge and not providing a theory of how knowledge or beliefs are manipulated [Newell, 1981]: "...there are not well-defined structural properties associated with access and augmentation." Hence, we may conclude that belief revision is a phenomenon not analyzable on the knowledge level.

However, the *theory of epistemic change* and the *logic of theory change* developed by Alchourrón, Gärdenfors, and Makinson [Alchourrón and Makinson, 1982, Alchourrón *et al.*, 1985, Makinson, 1985, Makinson, 1987, Gärdenfors, 1988, Gärdenfors, 1989], which will be described briefly in Section 2, show that at least some aspects of belief revision can be subject to a knowledge level analysis. Based on some *rationality postulates* any *epistemic change operation* should satisfy, various epistemic change operations on *deductively closed theories* are analyzed—some results of this investigation will be presented in detail in Section 3. This approach, which recently received a lot of interest in the AI community (e.g. [Gärdenfors and Makinson, 1988, Dalal, 1988]), suffers from some deficiencies, though. It is not clear how the results of the theory of epistemic change relate to belief revision as done in AI. Second, the theory of epistemic change does ignore what is usually called *reason maintenance*. These problems will be discussed in Section 4.

In Section 5, some approaches to belief revision in AI and database theory are presented. These will be analyzed by adapting the rationality predicates of the theory of epistemic change, which leads to the conclu-

*This paper describes research done at the Technical University of Berlin as part of ESPRIT project 311 and at IBM Germany as part of the LILOG project.

sion that these approaches satisfy most of the basic rationality postulates.

Based on that, in Section 6, an explicit reconstruction of *symbol-level* belief revision in terms of the theory of epistemic change is given—showing that a *knowledge level* analysis of belief revision techniques as developed in AI is indeed possible. It also shows that reason maintenance needs not to be integrated as a primitive notion in any theory of belief revision, but that it results as a side-effect of the reconstruction, contrary to the opinion most authors seem to have (cf. [Ginsberg, 1986, Gärdenfors, 1988]).

Finally, in Section 7, we will refine the reconstruction in order to satisfy all rationality postulates—leading to a belief revision strategy similar to the one used in the RUP system [McAllester, 1982].

2 The Theory of Epistemic Change

For the following discussion, we will assume a propositional language $\mathcal{L}$ containing propositions x, y, z and the standard sentential connectives ($\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$). Sets of propositions will be denoted by A, B, C. Furthermore, $\vdash$ shall denote classical propositional derivability[1] and Cn should be a function mapping sets of propositions to sets of propositions by applying $\vdash$, i.e.

$$Cn(A) \stackrel{\text{def}}{=} \{x \in \mathcal{L} \mid A \vdash x\} \tag{1}$$

Formalising Newell's notions of the knowledge level in this setting, sets of propositions A closed with respect to Cn (i.e. $A = Cn(A)$)—technically speaking *propositional theories*—can be identified with *knowledge-level knowledge bases* as argued in [Dietrich, 1986, Levesque and Brachman, 1986]. Arbitrary, finite set of propositions $B \subseteq \mathcal{L}$ can be identified with *symbol-level knowledge bases*.

In the theory of epistemic change [Gärdenfors, 1988], only knowledge-level knowledge bases are considered, which are called *belief sets*. *Epistemic change operations* on such belief sets are

Expansion: the monotonic addition of a belief with the requirement that the result is again a belief set (written $A + x$),

Contraction: the removal of a proposition from a belief set resulting in a new belief set (written $A \dot{-} x$),

Revision: incorporation of a new proposition into a belief set under the requirement that the result is a consistent belief set (written $A \dot{+} x$).

[1] The results presented in the following can be generalised to conservative extensions of propositional logics, provided they are compact and monotonic (cf. [Gärdenfors, 1988, p. 21–26]).

While expansion is a well-defined, unique operation, namely:

$$A + x \stackrel{\text{def}}{=} Cn(A \cup \{x\}) \tag{2}$$

the other two operations are problematical. An immediate criterion for them is that a belief set shall be changed minimally by an epistemic change operation (but cf. [Winslett, 1986]). However considering contraction, given a belief set B and a proposition x, in general there is no *unique greatest* belief set $C \subseteq B$ such that $C \not\vdash x$.

The problem of finding intuitively plausible change operations is approached by formulating sets of *rationality postulates* any epistemic change operation should satisfy. A set of such postulates for contraction can be given as follows (A a belief set, x, y propositions):

($\dot{-}1$) $A \dot{-} x$ *is a belief set* (*closure*);
($\dot{-}2$) $A \dot{-} x \subseteq A$ (*inclusion*);
($\dot{-}3$) *If* $x \notin A$ *then* $A \dot{-} x = A$ (*vacuity*);
($\dot{-}4$) *If* $\not\vdash x$, *then* $x \notin (A \dot{-} x)$ (*success*);
($\dot{-}5$) *If* $Cn(\{x\}) = Cn(\{y\})$ *then* $A \dot{-} x = A \dot{-} y$ (*preservation*);
($\dot{-}6$) $A \subseteq (A \dot{-} x) + x$ (*recovery*);
($\dot{-}7$) $(A \dot{-} x) \cap (A \dot{-} y) \subseteq A \dot{-} (x \wedge y)$;
($\dot{-}8$) *If* $x \notin A \dot{-} (x \wedge y)$, *then* $A \dot{-} (x \wedge y) \subseteq A \dot{-} x$.

Most of these postulates are straightforward. The *closure* postulate ($\dot{-}1$) tells us that we always get a belief set when applying $\dot{-}$ to a belief set and a proposition. The *inclusion* postulate ($\dot{-}2$) assures that when a proposition is removed, nothing previously unknown can enter into the belief set, setting an upper bound for any possible contraction operation. Postulate ($\dot{-}3$) takes care of one of the limiting cases, namely, that the proposition to be removed is not part of the belief set, while the next postulate ($\dot{-}4$) describes the effect of the other case. If the proposition to be removed is not a logically valid one, then the contraction operation will effectively remove it. The *preservation* postulate ($\dot{-}5$) assures that the syntactical form of the proposition to be removed will not effect the resulting belief set. Any two propositions which are logically equivalent shall lead to the same result. Finally, the *recovery* postulate ($\dot{-}6$) describes the lower bound of any contraction operation. The contracted belief set should contain enough information to recover all propositions deleted. Note that ($\dot{-}6$) together with ($\dot{-}1$)–($\dot{-}5$) entails the following conditional equation:

$$\text{If } x \in A \text{ then } A = (A \dot{-} x) + x \tag{3}$$

The two postulates ($\dot{-}7$) and ($\dot{-}8$) are less obvious and not as basic as the former ones—a reason for calling them "supplementary postulates." ($\dot{-}7$) states that retracting a conjunction should remove less information than retracting both conjuncts individually in

parallel, with (÷8) its conditional converse. Although this does not sound like a strong restriction, not all conceivable contraction operations satisfy it. In order to shed some more light on these supplementary postulates, it might be worthwhile to present some principles derivable from (÷7) and (÷8).[2] First, there is the following "factoring" condition:

$$A \div (x \wedge y) = \begin{cases} (A \div x) \cap (A \div y) & \text{or} \\ A \div x & \text{or} \\ A \div y \end{cases} \quad (4)$$

This condition is actually equivalent to (÷7) and (÷8) if a contraction operation already satisfies (÷1)-(÷6).

Another interesting property derivable from the supplementary postulates is an identity criterion for contracted belief sets:

$$\text{If } (x \rightarrow y) \in A \div y \text{ and } (y \rightarrow x) \in A \div x \text{ then } A \div y = A \div x \quad (5)$$

Turning now to revision, we note that there are two independent ways to characterise this operation. First, a set of rationality postulates for revision could be specified capturing the idea that a revised belief set should minimally differ from the original belief set, as done in [Alchourrón *et al.*, 1985]. Second, one could define the revision operation $A \dot{+} x$ by first contracting A with respect to $\neg x$ in order to avoid inconsistencies, and then expanding the result by x:

$$A \dot{+} x \stackrel{\text{def}}{=} (A \div \neg x) + x \quad (6)$$

This way of defining a revision operation was proposed by Levi [1977]. As it turns out, both ways of characterising revision are equivalent, as shown in [Alchourrón *et al.*, 1985]. Any revision operation satisfying the rationality postulates for revision could be generated by (6) and a contraction operation satisfying the contraction postulates and *vice versa*. What should be noted at this point is that in the case when $\vdash \neg x$ (and only in this case), the revised belief set will be inconsistent—which cannot be avoided, however.

Parallel to defining revision by contraction, we could try it the other way around—defining contraction in terms of revision:[3]

$$A \div x \stackrel{\text{def}}{=} (A \dot{+} \neg x) \cap A \quad (7)$$

This means, revision and contraction are interdefinable and it suffices to analyze one of these operations. Whether contraction or revision is taken as the basic one is mostly a matter of taste and philosophy (cf. [Makinson, 1985, Dalal, 1988]).

[2] Proofs for the principles (4) and (5) can be found in [Gärdenfors, 1988].

[3] Note that the intersection of two belief sets is again a belief set.

3 Constructing Contraction Functions

Using the rationality postulates, a number of possible contraction functions (and the associated revision functions) are studied and evaluated in [Alchourrón *et al.*, 1985] and [Gärdenfors, 1988].

All of these operations are defined using the family of maximal subsets not implying a given proposition, denoted by $A \downarrow x$ (pronounced "A less x"):

$$A \downarrow x \stackrel{\text{def}}{=} \{B \subseteq A \mid B \not\vdash x \text{ and if } B \subset C \subseteq A \text{ then } C \vdash x\} \quad (8)$$

Note that all elements of $A \downarrow x$ are again belief sets because of the maximality condition. Trying to construct contraction functions based on $A \downarrow x$, a first idea could be to take into account all possible outcomes of removing a proposition, and, since we do not have a measure of what is a better solution, to choose the intersection of the outcomes as the result of the contraction operation. If $A \downarrow x$ is empty—which can only happen if x is a logically valid proposition—A itself will be taken as the solution.

$$A \stackrel{f}{\div} x \stackrel{\text{def}}{=} \begin{cases} \bigcap (A \downarrow x) & \text{if } \not\vdash x \\ A & \text{otherwise} \end{cases} \quad (9)$$

This operation satisfies obviously most of the rationality postulates.[4]

Lemma 1 *Full meet contraction satisfies (÷1)-(÷5).*

In order to see that full meet contraction also satisfies (÷6), the following lemma is helpful.

Lemma 2 *Let A be a belief set, and let x be a proposition such that $x \in A$ and $\not\vdash x$, then*

$$A \stackrel{f}{\div} x = A \cap Cn(\{\neg x\}) \quad (10)$$

Applying this result to revision by using (6), it becomes obvious that full meet contraction is most probably not an operation one wants to use. Full meet contraction removes too much information.

Corollary 3 *For a revision operation defined by (6) and (9), for any x such that $\neg x \in A$ and $\not\vdash \neg x$ it holds that*

$$A \dot{+} x = Cn(\{x\}) \quad (11)$$

Nevertheless, full meet contraction satisfies all the rationality postulates for contraction.

Lemma 4 *Full meet contraction as defined by (9) satisfies (÷1)-(÷8).*

[4] In order to make the paper self-contained, I included proofs for all lemmas in this section in Appendix A.

Looking for a more reasonable contraction function, another way to contract a belief set could be to choose one of the elements in $A \downarrow x$—employing a *choice function* C—instead of using the intersection over all elements:

$$A \stackrel{\mathrm{m}}{-} x \stackrel{\mathrm{def}}{=} \begin{cases} C(A \downarrow x) & \text{if } \not\vdash x \\ A & \text{otherwise} \end{cases} \quad (12)$$

It is easy to see that this operation satisfies $(\dot{-}1)$–$(\dot{-}6)$, but the supplementary postulates are not satisfied unconditionally [Alchourrón and Makinson, 1982]. Ignoring this fact for the moment, let us try to characterize the result of such contraction operations. As it turns out, the contraction function defined by (12) generates belief sets which are far too large to be plausible.

Lemma 5 *Let A be a belief set with $x \in A$. Then for any proposition y:*

$$(x \vee y) \in A \stackrel{\mathrm{m}}{-} x \quad \text{or} \quad (x \vee \neg y) \in A \stackrel{\mathrm{m}}{-} x \quad (13)$$

This property has a rather counter-intuitive consequence for revision. Applying again (6), we get the following result.

Corollary 6 *Let $\dot{+}$ be a revision operation defined by using (12) and (6). Then, for any proposition x and belief set A with $\neg x \in A$:*

$$y \in A \dot{+} x \quad \text{or} \quad \neg y \in \dot{+} x \quad (14)$$

This means that by applying maxichoice revision to an arbitrary belief set, we get all of the sudden a *complete* belief set, provided that $\neg x \in A$. However, starting with an arbitrary belief set in which there may be no belief in some proposition z or its negation $\neg z$ and ending up with a belief set in which for all propositions z, either z or $\neg z$ is believed, is clearly something not desirable.

Viewing full meet contraction and maxichoice contraction as two extreme points, it might be worthwhile to explore the "middle ground" between them. Instead of chosing one element from $A \downarrow x$ or the entire family of belief sets, a subfamily of $A \downarrow x$ is used to generate the contracted belief set. For this purpose, let us assume a *selection function* $\mathcal{S}$ which selects a subset of $A \downarrow x$:

$$A \stackrel{\mathrm{p}}{-} x \stackrel{\mathrm{def}}{=} \begin{cases} \bigcap \mathcal{S}(A \downarrow x) & \text{if } \not\vdash x \\ A & \text{otherwise} \end{cases} \quad (15)$$

This contraction function, called *partial meet contraction*, unconditionally satisfies the basic postulates, which can be easily verified.

Lemma 7 *Any partial meet contraction operation $\stackrel{\mathrm{p}}{-}$ satisfies $(\dot{-}1)$–$(\dot{-}6)$.*

What is more interesting is that the converse holds as well. Any operation satisfying $(\dot{-}1)$–$(\dot{-}6)$ is a partial meet contraction [Gärdenfors, 1988, Theorem 4.13]. In order to satisfy the supplementary postulates, some restrictions on $\mathcal{S}$ must be imposed. Let us assume a "preference relation" $\sqsubseteq$ over all subsets of a belief set A independent of x such that for all $B, C \subseteq A$:

$$\text{if } B \in \mathcal{S}(A \downarrow x) \text{ and } C \in A \downarrow x \text{ then } C \sqsubseteq B \quad (16)$$

in which case the contraction function is called *relational contraction.*

Lemma 8 *Any relational contraction function satisfies $(\dot{-}1)$–$(\dot{-}7)$*

If the relation $\sqsubseteq$ is a transitive relation, the corresponding contraction function—called *transitively relational contraction*—satisfies all postulates.

Lemma 9 *Any transitively relational contraction function satisfies $(\dot{-}1)$–$(\dot{-}8)$.*

Furthermore, it is possible to show that any contraction function satisfying $(\dot{-}1)$–$(\dot{-}8)$ is a transitively relational contraction [Gärdenfors, 1988, Theorem 4.16].

4 Problems with the Approach

Although the results presented in the previous sections sound interesting and provide some insights into the problem of belief revision, it seems arguable whether the approach could be used in a computational context, as in AI or in the database field. The theory of epistemic change seems to capture only the *idealized* process of belief revision—ignoring some important problems of belief revision appearing in the "real world."

First of all, closed theories cannot be dealt with directly in a computational context because they are too large. At least, if we deal with them, we would like to have a finite representation (i.e. a finite axiomatization), and there seems to be no obvious way to derive a finite representation from a revised or contracted belief set in the general case.

Second, it seems to be preferable for pragmatic reasons to modify *belief bases*, i.e. finite sets of propositions, instead of belief sets, i.e. deductively closed theories. Propositions in belief bases usually represent something like facts, observations, rules, laws, etc., and when we are forced to change the belief set we would like to stay as close as possible to the original formulation of the finite base. In particular, when it becomes necessary to give up a proposition in the belief base, we would like to throw away the consequences of the retracted proposition if they are not supported otherwise, i.e. to perform *reason maintenance*. More formally, given a belief base B and propositions $x, y \in Cn(B)$ and assuming that a proposition y is removed from B to accomplish a contraction $Cn(B) \dot{-} x$, then

we expect that $z \notin (Cn(B) \dot{-} x)$ if y was responsible for z, i.e. $z \in Cn(B)$ but $z \notin Cn(B \setminus \{y\})$.

For instance, let a be the proposition "the device is ok," let b be the proposition "the output voltage is 5V" and let us assume we have the base $B = \{a, (a \rightarrow b)\}$, i.e. "the device is ok" and "if the device is ok then the output voltage is 5V." That means that from B we can infer that "the output voltage is 5V." Now, when we learn that the device is defect, then together with a we would like to get rid of b because the reason for the belief that the output voltage is 5V has vanished. This, however, cannot be easily accomplished by the approach described above. On the contrary, since the theory of the epistemic change formalizes the idea of keeping as much of the old propositions (in the belief set) as possible, it seems likely that b will be among the propositions in the contracted belief set since it does not contradict $\neg a$. Gärdenfors puts it in the following way [Gärdenfors, 1988, p. 67]:

> However, belief sets cannot be used to express that some beliefs may be *reasons* for other beliefs. (This deficiency was one of the motivations behind Doyle's TMS ...). And intuitively, when we compare degrees of similarity between different epistemic states, we want the structure of reasons or justifications to count as well.

Actually, viewing this property from a cognitive angle, it could be defended by the argument that the theory of epistemic change models what is called the *coherence theory* of belief revision [Gärdenfors, 1989]. This means that in the course of revising beliefs the main emphasis is to arrive at a new *coherent* set of beliefs, which may be interpreted as a *logically consistent* set of beliefs. Identifying and discarding *derived* beliefs when their justifications are undermined, on the other hand, is not viewed as essential in the *coherence theory*. It is argued that it is intellectually much too expensive to keep track of all justifications—a fact supported by empirical evidence.

Nevertheless, although this theory may be right in the general case, in a problem-solving context, as modeled in typical AI applications, we usually want *reason maintenance*—as e.g. in the toy example given above.

Summarizing, we see that *belief revision* and *reason maintenance* are not genuinely connected with each other, as it sometimes seems to be perceived in AI (cf. [Martins and Shapiro, 1988]). However, as will be shown, it is not necessary to add *reason maintenance* as a primitive notion to a theory of belief revision. Reason maintenance will result as a side-effect when we choose the "right" contraction operation.

5 Contracting Finite Bases

As spelled out in the previous section, there are good reasons to perform belief revision on belief bases—considering the propositions in the base as the basic beliefs. As a matter of fact, such operations were adopted in an analysis of update semantics for logical databases [Fagin *et al.*, 1983, Fagin *et al.*, 1986] and in modelling counterfactual reasoning [Ginsberg, 1986].

Basically, revision ($\tilde{+}$) and contraction ($\sim$) on a belief base B is defined in the following way:

$$B \sim x \stackrel{\text{def}}{=} \begin{cases} \bigvee_{C \in (B \downarrow x)} C & \text{If } \not\vdash x \\ B & \text{otherwise} \end{cases} \quad (17)$$

$$B \tilde{+} x \stackrel{\text{def}}{=} (B \sim \neg x) \wedge x \quad (18)$$

with $B \downarrow x$ being the same operation as defined by equation (8) without requiring that B is deductively closed. The guiding idea behind (17) and (18) is that we want to retain as many old propositions as possible, and if there are ties, we take the disjunction. Moreover, (18) is logically very similar to (6). Obviously, such change operations realize some form of reason maintenance, as one can see in (19).

$$\{a, a \rightarrow b\} \sim a \;=\; \{a \rightarrow b\} \not\vdash b \quad (19)$$

Based on (17) and (18), both Ginsberg [1986] and Fagin *et al.* [1983] consider more elaborated versions of contraction and revision, which distinguish between different kinds propositions in the belief base. For instance, Fagin *et al.* distinguish between *facts* and *integrity rules* in the belief base, and Ginsberg proposes to protect some propositions against retraction. We will ignore this issue here. However, one should note that such a construction is not qualitatively different from $\sim$ and $\tilde{+}$ [Nebel, 1989]. In particular, the results in this and the next section are valid for such operations.

In trying to relate $\sim$ to $\dot{-}$, we see that the rationality postulates presented in Section 2 cannot be applied immediately to $\sim$ since it does not operate on belief sets. However, it seems possible to adapt the postulates to belief bases by setting

$$A \stackrel{\text{def}}{=} Cn(B) \quad (20)$$

$$A \dot{-} x \stackrel{\text{def}}{=} Cn(B \sim x) \quad (21)$$

Thus, in a sense, we view $\sim$ as an implementation of $\dot{-}$.

Lemma 10 *Under the assumption of (20) and (21), $\sim$ satisfies ($\dot{-}$1)-($\dot{-}$5).*

Proof: ($\dot{-}$1) holds trivially because of (21). If $\not\vdash x$, ($\dot{-}$2) is satisfied because

$$Cn(\bigvee_{C \in (B \downarrow x)} C) \;=\; \bigcap_{C \in (B \downarrow x)} Cn(C) \quad (22)$$

and for all C : $Cn(C) \subseteq Cn(B)$. If $\not\vdash x$, ($\dot{-}$2) is satisfied as well since the belief base is not changed. ($\dot{-}$3) is

satisfied because if $B \not\vdash x$, then $B \downarrow x = \{B\}$. $(\dot{-}4)$ holds because for all $C \in (B \downarrow x)$ we have $C \not\vdash x$ and, hence, $\bigvee C \not\vdash x$. Finally, $(\dot{-}5)$ holds since for the determination of $B \downarrow x$ the syntactic form of x does not matter. ■

Unfortunately, however, the recovery postulate is not satisfied. For instance, we have

$$Cn((\{a, a \rightarrow b\} \sim b) \cup \{b\}) \not\supseteq Cn(\{a, a \rightarrow b\}) \quad (23)$$

Trying to find the reason for this unsatisfying behavior, one notes that even the weakest possible contraction function on belief sets—full meet contraction—generates belief sets such that (Lemma 2)

$$A \dot{-} x = A \cap Cn(\{\neg x\})$$

which is sufficient for restoring the original belief set as we have seen in Lemma 4.

Adding a finite conjunct, logically equivalent to $A \cap Cn(\{\neg x\})$, to the outcome of $\sim$ leads to a new contraction function which has the desired property:

$$B \dot{\sim} x \stackrel{\text{def}}{=} \begin{cases} (\bigvee_{C \in (B \downarrow x)} C) \wedge (B \vee \neg x) & \text{if } \not\vdash x \\ B & \text{otherwise} \end{cases} \quad (24)$$

Lemma 11 *Under the assumption of (20) and (21), $\dot{\sim}$ satisfies $(\dot{-}1)$-$(\dot{-}6)$.*

Proof: The satisfaction of $(\dot{-}1)$, $(\dot{-}3)$, and $(\dot{-}5)$ can be shown with the arguments used in the proof of Lemma 10. That $(\dot{-}2)$ holds becomes obvious by observing that $Cn(B \vee \neg x)$ and $Cn(\bigvee C)$ are both subsets of $Cn(B)$, and, hence, the set of consequences of their unions can only be a subset of $Cn(B)$, either. $(\dot{-}4)$ holds because for the added conjunct we have $(B \vee \neg x) \not\vdash x$ and no $C \in B \downarrow x$ implies x. Finally, $(\dot{-}6)$ holds since $Cn(B \dot{\sim} x)$ contains $Cn(B) \cap Cn(\{\neg x\})$, which is sufficient for recovery as shown in the proof of Lemma 4. ■

Actually, if revision is the only operation of interest, it does not make a difference whether we employ $\dot{\sim}$ or $\sim$. The revision operation $\tilde{+}$ gives identical results (*wrt* Cn) regardless of whether $\dot{\sim}$ or $\sim$ is used.

Theorem 12 *The operations $\dot{\sim}$ and $\sim$ are revision-equivalent wrt $\tilde{+}$ as defined by (18), i.e.*

$$Cn((B \dot{\sim} \neg x) \wedge x) = Cn((B \sim \neg x) \wedge x) \quad (25)$$

Proof: In the limiting cases when $\vdash \neg x$ or $\neg x \notin Cn(B)$, $\sim$ and $\dot{\sim}$ give the same results trivially. For the principal case, $\not\vdash \neg x$ and $\neg x \in Cn(B)$, we have:

$$\begin{aligned} Cn((B \dot{\sim} \neg x) \wedge x) &= \\ &= Cn((\bigvee_{C \in (B \downarrow \neg x)} C) \wedge (B \vee x) \wedge \{x\}) \\ &= Cn((\bigvee_{C \in (B \downarrow \neg x)} C) \wedge x) \\ &= Cn(B \sim \neg x) + x \end{aligned}$$

■

This result might raise the question of the value of the recovery postulate—a problem discussed in [Makinson, 1987]. Despite the fact that it is necessary to establish some of the theoretical results cited in Section 3, it has some practical value, I believe. In a setting where revision and contraction are equally important, as in the case of database updates (cf. [Fagin *et al.*, 1986]), we had better use $\dot{\sim}$ instead of $\sim$. Otherwise, more information is lost than intended. In particular, we might be unable to *undo* a contraction operation.

Before we now go on trying to verify that the supplementary postulates are satisfied by $\dot{\sim}$, we will try to establish a connection between belief-base and belief-set contraction. In [Ginsberg, 1986], as well as in [Fagin *et al.*, 1983], some thoughts are devoted to the issue of modifying closed theories, i.e. belief sets. However, by considering (22) and permitting infinite disjunctions, it becomes quickly obvious that in the limiting case when the belief base is a belief set, $\sim$ is identical to full meet contraction. This kind of contraction is rather useless, however, as demonstrated by Lemma 2 and Corollary 3. Thus, Fagin *et al.* [1983] and Ginsberg [1986] conclude that belief revision is a phenomenon which can only be modeled if the *syntactic representation* of a belief set (its belief base) is taken into account. A knowledge-level analysis of belief revision seems to be impossible.

Addressing this problem, Ginsberg [1986, Section 8.1] proposes to incorporate *reason-maintenance* information into the logic by using derivations as truth-values. This proposal leads to the desired results, i.e. changes of belief sets (in the reason-maintenance style logic) are identical to changes of a finite belief bases. However, this approach does not shed too much light onto the relation between modifications of belief sets and modifications of finite belief bases.

6 Base Contraction Viewed as Partial Meet Contraction

Reconsidering the arguments from above, we note that we are not interested in the particular syntactical form of a belief base, but we regard the propositions in the belief base as somehow more important than any derived propositions. In particular, given two belief bases B and B' such that for all $x \in B$ there exists $x' \in B'$ with $Cn(\{x\}) = Cn(\{x'\})$ and *vice versa*, it is immediate that $Cn(B \dot{\sim} y) = Cn(B' \dot{\sim} y)$. This means that it is not the syntactical form of a belief base we are interested in, but the "logical force" of the propositions in the base. Using this idea, it is tempting to define a selection function which selects elements from $(A \downarrow x)$ which contain maximal subsets of B, a set of

propositions considered as "interesting":

$$\mathcal{S}_B(A\downarrow x) \stackrel{\text{def}}{=} \{C \in (A\downarrow x) | \forall C' \in A\downarrow x C' \cap B \not\supset C \cap B\}$$

Based on this selection function, we can define a partial meet contraction on belief sets which has the property of being identical to $\dot{-}$ (*wrt* Cn). In order to show this, the next lemma proves to be helpful.

Lemma 13 *Let A be a belief set and S be any subset of A such that $S \not\vdash x$. Then*

$$\bigcap\{C \in (A\downarrow x) |\ S \subseteq C\} = Cn(S \cup (A \cap Cn(\{\neg x\})))$$

Proof: "$\supseteq$": First, by Lemma 2 we know that any intersection over a subfamily of $A \downarrow x$ must contain $(A \cap Cn(\{\neg x\}))$. Second, since all C in the subfamily chosen contain S, the LHS contains S. Finally, because the intersection over any subfamily of $A \downarrow x$ is a belief set, the LHS is a belief set containing S and $(A \cap Cn(\{\neg x\}))$ and, hence, the right hand side.

"$\subseteq$": Assume the contrary, i.e. there is a y such that $y \in$ LHS and $y \notin$ RHS. Using set theory and the properties of Cn, we can transform the RHS in the following way:

$$\begin{aligned} Cn(S \cup (A \cap Cn(\{\neg x\}))) = \\ &= Cn((S \cup A) \cap (S \cup Cn(\{\neg x\}))) \\ &= Cn(A \cap Cn(S \cup Cn(\{\neg x\}))) \\ &= A \cap Cn(S \cup \{\neg x\}) \end{aligned}$$

Since $y \in A$ because of our assumption, we must have $y \notin Cn(S \cup \{\neg x\})$ and, in particular, $\neg x \not\vdash y$. Using this, we can derive $(x \vee \neg y) \not\vdash x$, following the same line of arguments as in the proof of Lemma 2. By that and the observation that $y \notin Cn(S)$, we can conclude that $x \notin Cn(S \cup \{(x \vee \neg y)\})$. Since adding y to this set would lead to the derivation of x, there must be a set $E \supseteq S \cup \{(x \vee \neg y)\}$ with $y \notin E$ and $E \in (A \downarrow x)$, which means that y cannot be a member of all sets in $A \downarrow x$ which contain S, and we have a contradiction. ■

Based on this lemma, we can establish the connection between contractions on belief bases and contractions on belief sets.

Theorem 14 *Contraction of finite premise sets B using $\dot{-}$ is identical (wrt $\vdash$) to a partial meet contraction $\stackrel{\mathrm{p}}{-}$ defined by the selection function $\mathcal{S}_B$, i.e.*

$$Cn(B \dot{-} x) \quad = \quad Cn(B) \stackrel{\mathrm{p}}{-} x \qquad (26)$$

Proof: In the limiting cases when $\vdash x$ or $x \notin Cn(B)$ the result is immediate. For the principal case, we note that

$$\mathcal{S}_B(Cn(B)\downarrow x) = \bigcup_{D \in (B\downarrow x)} \{C \in (Cn(B)\downarrow x) | D \subseteq C\} \qquad (27)$$

Applying Lemma 13, it follows that

$$\begin{aligned} Cn(B) \stackrel{\mathrm{p}}{-} x = \\ &= \bigcap \mathcal{S}_B(Cn(B)\downarrow x) \\ &= \bigcap_{D \in (B\downarrow x)} Cn((D) \cup (Cn(B) \cap Cn(\{\neg x\}))) \\ &= Cn((\bigcap_{D \in (B\downarrow x)} Cn(D)) \cup (Cn(B) \cap Cn(\{\neg x\}))) \\ &= Cn((\bigvee_{D \in (B\downarrow x)} Cn(D)) \wedge (B \vee \neg x)) \\ &= Cn(B \dot{-} x) \end{aligned}$$

■

Thus, contrary to the assumptions spelled out in [Dietrich, 1986, Fagin *et al.*, 1983, Fagin *et al.*, 1986, Ginsberg, 1986], revision and contraction on finite belief bases are not qualitatively different from epistemic change operations on deductively closed belief sets. The finite case can be modeled without any problem by a particular selection function. Viewed from a knowledge-level perspective, the only additional information needed for belief revision is a preference relation on sets of propositions. It should be noted, however, that the construction did not lead to an epistemic change function which satisfies all rationality postulates.

Theorem 15 *Any partial meet contraction using $\mathcal{S}_B$ satisfies the rationality postulates* $(\dot{-}1)$-$(\dot{-}7)$.

Proof: Because of Lemma 7 and the fact that $\stackrel{\mathrm{p}}{-}$ constructed by $\mathcal{S}_B$ is a partial meet contraction, $(\dot{-}1)$-$(\dot{-}6)$ are satisfied. That $(\dot{-}7)$ is satisfied follows from Lemma 8 and the fact that $\mathcal{S}_B$ satisfies (16), if $\sqsubseteq$ is defined as:

$$X \sqsubset Y \quad \textit{iff} \quad X \cap B \not\supset Y \cap B \qquad (28)$$

■

Since $\mathcal{S}_B$ is not a transitively relational selection function, $(\dot{-}8)$ is not satisfied in general. In order to give an example where $(\dot{-}8)$ is violated, let us assume that $\mathcal{S}_B$ is used to define the partial meet contraction $\stackrel{\mathrm{p}}{-}$ and that

$$B \quad = \quad \{a, b \wedge c, a \wedge b \wedge d, a \wedge d\}$$

Setting, $x = (a \wedge c)$ and $y = (b \wedge d)$, we see that

$$x \notin Cn(B) \stackrel{\mathrm{p}}{-} (x \wedge y)$$

but

$$Cn(B) \stackrel{\mathrm{p}}{-} (x \wedge y) \not\subseteq Cn(B) \stackrel{\mathrm{p}}{-} x$$

because we have the following relationships:

$$\begin{aligned} (a \wedge c) \quad &\notin \quad Cn(B) \stackrel{\mathrm{p}}{-} ((a \wedge c) \wedge (b \wedge d)) \\ a \quad &\in \quad Cn(B) \stackrel{\mathrm{p}}{-} ((a \wedge c) \wedge (b \wedge d)) \\ a \quad &\notin \quad Cn(B) \stackrel{\mathrm{p}}{-} (a \wedge c) \end{aligned}$$

As can be verified, the factoring condition (4) is violated, as well. So what? Does this result imply that $\stackrel{p}{-}$ defined by using $\mathcal{S}_B$ and, hence, $\dot{\sim}$ is not a reasonable contraction operation? Actually, it seems to make a lot more sense than $\stackrel{f}{-}$.

The disadvantage of not having $(\dot{-}8)$ is actually very subtle. One consequence is that a revision operations based on $\dot{\sim}$ violate the respective postulate for revision, which is needed to derive an identity criterion for revised belief sets similar to (5):

$$\text{If } x \in A \dot{+} y \text{ and } y \in A \dot{+} x \text{ then } A \dot{+} y = A \dot{+} x \qquad (29)$$

This criterion in turn is is similar to a principle Stalnaker [1968] postulated for the interpretation of counterfactual conditionals on "neighboring possible worlds." However, it seems to be difficult to come up with an example demonstrating that the violation of this principle leads to obviously counter-intuitive results.

7 Maxichoice Contraction on Belief Bases

As we noted above, we could achieve satisfaction of $(\dot{-}8)$ if the ordering $\sqsubseteq$ is transitive. Embedding the partial order defined by set-inclusion in a total ordering would certainly help. This can be achieved by starting from an arbitrary total ordering on the propositions in a belief base, for instance.[5] Thus, let us assume such an ordering $\leq$ on the propositions of a belief base B. Furthermore, define $\sqsubseteq'$ on 2^B:

$$X \sqsubseteq' Y \quad \text{iff} \quad \forall x \in (X \setminus Y)\, \exists y \in (Y \setminus X) : x \leq y \qquad (30)$$

This means, in case when X and Y are incomparable by set-inclusion, we assume Y to be larger than X by $\sqsubseteq'$ iff there is an element in Y which is larger by $\leq$ than any element in X which is not in Y. Based on that, let us define a selection function as follows:

$$\mathcal{S}_{B,\leq}(A \downarrow x) \stackrel{\text{def}}{=} \{C \in (A \downarrow x) \mid \forall C' \in (A \downarrow x) : C' \cap B \sqsubseteq' C \cap B\}$$

Theorem 16 *A partial meet contraction defined by using $\mathcal{S}_{B,\leq}$ satisfies all rationality postulates.*

Proof: Obviously, $\sqsubseteq'$ is a transitive relation, and, thus, the partial meet contraction defined by $\mathcal{S}_{B,\leq}$ is a transitively relational contraction. ■

Another interesting point about $\mathcal{S}_{B,\leq}$ is that it is similar to a maxichoice contraction on belief bases.

[5] Note that such a construction is fundamentally different from the notion of epistemic entrenchment as introduced in [Gärdenfors, 1988, Gärdenfors and Makinson, 1988].

Lemma 17 *For partial meet contractions defined by using $\mathcal{S}_{B,\leq}$, for all x with $\not\vdash x$, there is an $E \in (Cn(B) \downarrow x)$ such that*

$$Cn(B) \stackrel{p}{-} x = Cn(E \cup (Cn(B) \cap Cn(\{\neg x\}))) \qquad (31)$$

Proof: The main point is that there is exactly one element in $E \in (B \downarrow x)$ such that for all $C \in (B \downarrow x)$ we have $C \sqsubseteq' E$. Applying Lemma 13, we get the desired result. ■

From this we can conclude two things. First, maxichoice contraction on belief bases does not have the undesirable result as the same operation applied to belief sets. Second, considering the rationality postulates, it is more "rational" to chose one alternative of $(B \downarrow x)$ than taking the disjunction of all alternatives. Furthermore, from a practical point of view, this strategy has the advantage that the belief base is not cluttered with disjunctions, but it simply shrinks.

Summarizing, although it might seem arbitrary to choose one maximal consistent set during a contraction, it is despite its practical value justified because it is at least as "rational" as using the disjunctions. Thus, for example, the *truth-maintenance system* RUP [McAllester, 1982], which implements this strategy for belief revision, could be characterized as a "fully rational belief revision system" (modulo its inferential incompleteness).

8 Conclusion and Outlook

We have shown that belief revision, as exercised in many applications in AI, is not an activity which can only be analyzed on the symbol level. Employing the theory of epistemic change we demonstrated how to reconstruct symbol-level belief revision in the theory of epistemic change—resulting in a knowledge-level analysis of some aspects of belief revision. In particular, we have shown that *reason maintenance* is a symbol-level notion, which although not present in the theory of epistemic change, appears as a side-effect. Furthermore, analyzing the approaches, we noted that chosing one maximal consistent subset of a belief base seems to be more rational than taking disjunctions of all maximal consistent sets—considering the rationality postulates.

However, a number of issues remain unresolved—of course. For instance, iterated contractions were ignored because they present serious problems. One has to provide update operations for the preference relation among propositions. Furthermore, implausible contraction operations, such as $\{a \wedge b\} \sim a = \emptyset$, were ignored. Choosing the "right" form of the premises seems to be one of the central tasks before any kind of belief revision can be applied.

Acknowledgments

I am grateful to Alex Borgida, Mukesh Dalal, and Karl Schlechta who provided me with a number of hints and ideas concerning the issues discussed in this paper. I would also like to express my thanks to Peter Gärdenfors, who sent me copies of his papers, made available the manuscript of his book, and provided me with some helpful comments an earlier version of this paper.

A Proofs of Lemmas of Section 3

Lemma 1 *Full meet contraction satsfies (÷1)-(÷5).*

Proof: (÷1) is satisfied because the intersection of belief sets is a belief set. (÷2) is satisfied because for all $E \in (A \downarrow x)$, $E \subseteq A$. (÷3) holds because $A \downarrow x = \{A\}$, if $A \not\vdash x$. (÷4) holds since for all $E \in (A \downarrow x)$, $E \not\vdash x$, if $\not\vdash x$. (÷5) is satisfied since the syntactical form of x in $A \downarrow x$ is irrelevant. ■

Lemma 2 *Let A be a belief set, and let x be a proposition such that $x \in A$ and $\not\vdash x$, then*

$$A \stackrel{f}{-} x = A \cap Cn(\{\neg x\})$$

Proof: First, consider the case when $y \in A$ and $\neg x \vdash y$. Now assume that $y \notin A \stackrel{f}{-} x$. That means that there is a set $E \in A \downarrow x$ such that $y \notin E$. Because of the maximality condition on all such sets, we know that $x \in Cn(E \cup \{y\}))$. Using contraposition on our premise $\neg x \vdash y$, we get $\neg y \vdash x$ and hence $x \in Cn(E \cup \{\neg y\})$. Together with the previous result, we have $x \in Cn(E \cup \{(y \vee \neg y)\}) = Cn(E)$ and a contradiction.

For the case $y \in A$ and $\neg x \not\vdash y$, we know by contraposition that $\neg y \not\vdash x$ and hence $x \vee \neg y \not\vdash x$. Because of the maximality of the sets in $A \downarrow x$, there are two sets E', E'' with $y \in E'$ and $(x \vee \neg y) \in E''$, but there can be no set which includes both because $x \in Cn(\{y, (x \vee \neg y)\})$, and thus $y \notin \bigcap(A \downarrow x)$. ■

Corollary 3 *For a revision operation defined by (6) and (9), for any x such that $\neg x \in A$ and $\not\vdash \neg x$ it holds that*

$$A \dot{+} x = Cn(\{x\})$$

Proof: Using (6) and (10) leads to:

$$\begin{aligned} A \dot{+} x &= Cn((A \cap Cn(\{x\})) \cup \{x\}) \\ &= Cn((A \cup \{x\}) \cap (Cn(\{x\}) \cup \{x\})) \\ &= Cn(Cn(A \cup \{x\}) \cap Cn\{x\}) \end{aligned}$$

Since we assumed $A \vdash \neg x$, we know $Cn(A \cup \{x\}) = \mathcal{L}$, and, thus, the result is $Cn(\{x\})$. ■

Lemma 4 *Full meet contraction as defined by (9) satisfies (÷1)-(÷8).*

Proof Sketch: Since (÷1)-(÷5) are obvious, we will focus on (÷6)-(÷8). These are satisfied trivially for the two limiting cases $\vdash x$ and $x \notin A$. That (÷6) holds in the principal case, $x \in A$ and $\not\vdash x$, becomes obvious when substituting the right hand side of equation (10) for $A \div x$ in (÷6), which leads to:

$$A \subseteq Cn((A \cap Cn(\{\neg x\})) \cup \{x\}) \qquad (32)$$

Now, because for any $y \in A$ we know that $(y \vee \neg x) \in Cn(\{\neg x\})$ and that this together with x implies y, the right hand side of (32) is clearly a superset of the left hand side. Furthermore, using Lemma 2, it can be easily derived that (÷7) (it can even be strengthened to equality) and (÷8) hold as well. ■

Lemma 5 *Let A be a belief set with $x \in A$. Then for any proposition y:*

$$(x \vee y) \in A \stackrel{m}{-} x \quad \text{or} \quad (x \vee \neg y) \in A \stackrel{m}{-} x$$

Proof: In the limiting cases when $\vdash x$, the lemma holds trivially. In the other case, we know that $y \not\vdash x$ or $\neg y \not\vdash x$. Thus, because of the maximality of the elements of $A \downarrow x$, either y or $\neg y$ is in $A \stackrel{m}{-} x$. Since for any z: $y \vdash (y \vee z)$, the lemma holds. ■

Corollary 6 *Let $\dot{+}$ be a revision operation defined by using (12) and (6). Then, for any proposition x and belief set A with*

$$y \in A \dot{+} x \quad \text{or} \quad \neg y \in \dot{+} x$$

Proof: Expanding $A \stackrel{m}{-} \neg x$ by x leads by Lemma 5 for all propositions y to

$$\begin{aligned} ((\neg x \vee y) \wedge x) &\in (A \stackrel{m}{-} \neg x) + x \quad \text{or} \\ ((\neg x \vee \neg y) \wedge x) &\in (A \stackrel{m}{-} \neg x) + x \end{aligned}$$

from which the desired result is immediate. ■

Lemma 7 *Any partial meet contraction operation $\stackrel{p}{-}$ satisfies (÷1)-(÷6).*

Proof Sketch: (÷1)-(÷5) can be easily verified. (÷6) holds because $A \div x$ always contains $A \cap Cn(\{\neg x\})$, which is sufficient for recovery. ■

Lemma 8 *Any relational contraction function satisfies (÷1)-(÷7)*

Proof: (÷1)-(÷6) follow from Lemma 7. If $\vdash x$, $\vdash y$, $x \notin A$, or $y \notin A$, the proof is immediate. For the other cases, we have to show that

$$\bigcap \mathcal{S}(A \downarrow x) \cap \bigcap \mathcal{S}(A \downarrow y) \subseteq \bigcap \mathcal{S}(A \downarrow (x \wedge y))$$

This could be done by showing that

$$(\mathcal{S}(A \downarrow x) \cup \mathcal{S}(A \downarrow y)) \supseteq \mathcal{S}(A \downarrow (x \wedge y)$$

Assume the contrary, i.e. there is an $E \in$ RHS, but $E \notin$ LHS. Since we cannot have $x, y \in E$, assume *wlg* $x \notin E$. Now it could be the case that $E \notin$ LHS because $E \notin (A \downarrow x)$. However, since $x \notin E$, there must be $D \in (A \downarrow x)$ with $E \subset D$. Furthermore, because $(x \wedge y) \notin D$, there is an $F \in (A \downarrow (x \wedge y))$ with $E \subset D \subseteq F$, which is a contradiction. This means $E \in (A \downarrow x)$. Accepting this, we must have $D \in (A \downarrow x)$ with $D \not\subseteq E$. Because D is maximal, either $y \in D$ or $x \in Cn(D \cup \{y\})$. Thus, $D \in (A \downarrow (x \wedge y))$, and because of relationality, the premise $E \in$ RHS is contradicted. ■

Lemma 9 *Any transitively relational contraction function satisfies (÷1)-(÷8).*

Proof: (÷1)-(÷7) follows from Lemma 8. Similar to the proof above, for the principal case, we will show that when $x \notin \bigcap \mathcal{S}(A \downarrow (x \wedge y))$, then

$$\mathcal{S}(A \downarrow (x \wedge y)) \supseteq \mathcal{S}(A \downarrow x)$$

Assume the contrary, i.e. $E \in$ RHS but $E \notin$ LHS. Because E is maximal, either $y \in E$ or $x \in Cn(E \cup \{y\})$, and hence $E \in (A \downarrow (x \wedge y))$. Since $E \notin$ LHS, there is $D \in (A \downarrow (x \wedge y))$ with $D \not\subseteq E$. Because of our premise $x \notin \bigcap \mathcal{S}(A \downarrow (x \wedge y))$, there must be at least one $F \in \mathcal{S}(A \downarrow (x \wedge y))$ with $x \notin F$, which is also an element of $(A \downarrow x)$. Thus $D \sqsubseteq F$ and $F \sqsubseteq E$, and because of transitivity $D \sqsubseteq E$, which is a contradiction. ■

References

[Alchourrón and Makinson, 1982] Carlos E. Alchourrón and David Makinson. On the logic of theory change: contraction functions and their associated revision functions. *Theoria* **48**: 14–37, 1982.

[Alchourrón *et al.*, 1985] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. *The Journal of Symbolic Logic* **50**(2): 510–530, June 1985.

[Dalal, 1988] Mukesh Dalal. Investigations Into a Theory of Knowledge Base Revision: Preliminary Report. *Proc. AAAI-88*, , pp. 475–479. Saint Paul, Minn., 1988.

[de Kleer, 1986] Johan de Kleer. An Assumption-based TMS. *Artificial Intelligence* **28**(2): 127–162, March 1986.

[Diettrich, 1986] Thomas G. Diettrich. Learning at the Knowledge Level. *Machine Learning* **1**: 287–316, 1986.

[Doyle, 1979] Jon Doyle. A Truth Maintenance System. *Artificial Intelligence* **12**(3): 231–272, 1979.

[Fagin *et al.*, 1983] Ronald Fagin, Jeffrey D. Ullman, and Moshe Y. Vardi. On the Semantics of Updates in Databases. *Proc. Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pp. 352–365. Atlanta, 1983.

[Fagin *et al.*, 1986] Ronald Fagin, Gabriel M. Kuper, Jeffrey D. Ullman, and Moshe Y. Vardi. Updating Logical Databases. *Advances in Computing Research* **3**: 1–18, 1986.

[Gärdenfors, 1988] Peter Gärdenfors. *Knowledge in Flux—Modeling the Dynamics of Epistemic States*. Cambridge, Mass.: MIT Press, 1988.

[Gärdenfors, 1989] Peter Gärdenfors. The Dynamics of Belief Systems: Foundations vs. Coherence Theories. Lund, Sweden: Department of Philosophy, Lund University, in preparation, January 1989.

[Gärdenfors and Makinson, 1988] Peter Gärdenfors and David Makinson. Revision of Knowledge Systems Using Epistemic Entrenchment. To appear in *Proc. 2nd Workshop on Nonmonotonic Reasoning*, Munich, West Germany, 1988.

[Ginsberg, 1986] Matthew L. Ginsberg. Counterfactuals. *Artificial Intelligence* **30**(1): 35–79, 1986.

[Levesque and Brachman, 1986] Hector J. Levesque and Ronald J. Brachman. Knowledge Level Interfaces to Information Systems. In M. L. Brodie and J. Mylopoulos (eds.), *On Knowledge Base Mangement Systems*, pp. 13–34. Berlin: Springer, 1986.

[Levi, 1977] Isaac Levi. Subjunctives, dispositions and chances. Synthese **34**: 423–455, 1977.

[Lewis, 1973] David K. Lewis. *Counterfactuals*. Cambridge, Mass.: Harvard University Press, 1973.

[Makinson, 1985] David Makinson. How to give it up: A Survey of Some Formal Aspects of Theory Change. *Synthese* **62**: 347–363, 1985.

[Makinson, 1987] David Makinson. On the Status of the Postulate of Recovery in the Logic of Theory Change. *Journal of Philosophical Logic* **16**: 383–394, 1987.

[Martins and Shapiro, 1988] João P. Martins and Stuart C. Shapiro. A Model for Belief Revision. *Artificial Intelligence* **35**(1): 25–79, May 1988.

[McAllester, 1982] David A. McAllester. Reasoning Utility Package User's Manual. AI Memo 667, Cambridge, Mass.: AI Laboratory, MIT, 1982.

[Nebel, 1989] Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*. Ph.D. thesis, in preparation, 1989.

[Newell, 1981] Allen Newell. The Knowledge Level. The Presidential Address, AAAI-80, Stanford, Cal. *The AI Magazine* **2**(2): 1–20, Summer 1981.

[Reiter, 1987] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence* **32**(1): 57–95, April 1987.

[Stalnaker, 1968] Robert C. Stalnaker. A Theory of Conditionals. In N. Rescher (ed.), *Studies in Logical Theory*. Oxford: Oxford University Press, 1968.

[Winslett, 1986] Marianne S. Winslett. Is Belief Revision Harder than You Thought?. *Proc. AAAI-86*, pp. 421–427. Philadelphia, Pa., August 1986.

Defaults and probabilities; extensions and coherence

Eric Neufeld*
School of Computer Science
University of New Brunswick
Fredericton, New Brunswick, CANADA, E3B 5A3

Abstract

Default logic offers a compelling solution to the problem of reasoning in commonsense domains: in uncertain domains, intelligent creatures use defaults to "jump" to certain conclusions, given no information to the contrary. However, no one in that camp has offered a definition of defaults that is testable, at least in principle. This shortcoming is critical in light of apparent paradoxes and ever-increasing complexity of the formalisms.

We offer a simple solution to the commonsense reasoning problem based on a view that probability is a theory of *sound approximate argument*. From some probabilistic inequalities and knowledge about conditional independence, we show it is possible to make many interesting inferences about *shifts* in belief based on probabilistic associations without numeric probability distributions. We can also use this knowledge to construct *coherent* models of the world based on a handful of observations. Most importantly, we can test the correctness of a particular representation by performing a statistical experiment. This challenges the claim that probability is "epistomologically inadequate" for reasoning in AI domains. In fact, the criterion of testability challenges the epistomological adequacy of other approaches.

1 Introduction

Default logicians claim that we "jump" to certain conclusions despite the fact most general rules have numerous counterexamples. The classic example is the "birds fly" problem where we believe birds fly yet emus don't. Modelling this kind of reasoning has challenged researchers for decades.

It is well-known that deductive use of first order logic does not work: either there are no emus or the knowledge is inconsistent. Many extensions of first order logic have been proposed [Reiter, 1980, McCarthy, 1986, Moore, 1985, Poole *et al.*, 1987] to overcome this. The formalisms go by many names, but in light of certain equivalence proofs [Grosof, 1985, Konolige, 1987], it is reasonable to refer to them as default logics. A unifying theme in all these papers is the idea that we must be able to *accept* defeasible conclusions without sacrificing the attractive semantics of first order logic.

A glance at recent literature raises at least three questions:

1. The complexity of the solutions seems to have by far outstripped the apparent simplicity of the examples [Etherington, 1988].
2. There are many different default logics. Representing the "birds fly" default in the obvious way in different formalisms yields slightly different answers. This raises the important question: just exactly *what* is a default? To the best of our knowledge, no one within the camp of default logic has offered a clear-cut answer to this problem. Specifically, how could we perform an experiment to verify whether a sentence α is a default or not?
3. It is questionable whether it is worthwhile even trying to find a logical solution to the default inference problem. Some results suggest there are problems with the very foundations of default inference. It has been believed for some time that the notion of a default logic must contain a lottery paradox which arises when we accept sentences with probability above a certain threshold. It appears that other default logics contain a variation of Simpson's paradox. We illustrate these problems in detail later.

Here we argue that there is indeed a problem with some of the default logics but we need not abandon logic. However, we need not *confine* ourselves to logic either: some basic ideas from probability theory cast light on why problems exist with default logic.

*Research supported at various stages by a Natural Science and Engineering Research Council of Canada postgraduate scholarship, the Institute for Computer Research at the University of Waterloo and the University of New Brunswick.

The same notions provide simple solutions to many of the problems. Far from sacrificing semantics, we strengthen the meaning of our representation, for we believe this to be one of the first *testable* accounts of commonsense reasoning.

After presenting a set of problems default logic must address, we offer a constructive probabilistic solution. Not only is the system capable of answering queries about flying birds and grey elephants, it is capable of constructing *coherent* models of the world based on a few observations. We illustrate an interesting probabilistic relationship between diagnosis and default inference. We see also that there is a correspondence between construction of coherent models and plan recognition.

We emphasize that no numerical probabilities are required, but we must believe that they exist and have certain properties.

2 Some problems of default logic

It is impossible to precisely say "default logic" contains this or that paradox, because there are many default logics and many problems only partially solved. Perhaps we should say these are problems every default logic must address.

2.1 The multiple extension problem

Poole et al [1986] represent the "birds fly" problem with the following sets $\mathcal{F}$ of facts and $\mathcal{D}$ of defaults:

$$\mathcal{F} = \{emu \rightarrow bird\},$$
$$\mathcal{D} = \{emu \rightarrow \neg fly,\ bird \rightarrow fly\}.$$

(We assume there are some flying emus, that is, we allow exceptions to exceptions.) For any goal g, an *explanation* of g is a subset d of $\mathcal{D}$ such that

$\mathcal{F} \cup d \models g$, and
$\mathcal{F} \cup d$ is consistent.

But then we can construct consistent explanations that emus don't fly and that emus do fly! This is the "multiple extension" [Hanks and McDermott, 1986] or "wrong answer" problem.

Many solutions exploit "specificity", that is, selecting the answer that depends on the "most specific" knowledge. In most presentations, *emu* would be more "specific" than *bird*. So long as all emus are birds, "most specific" corresponds to implication or the subset relation and we are all right. But some default logicians feel that if $emu \rightarrow bird$ were a default, we should conclude $\neg fly$ from $bird \wedge emu$. (For example, consider the "unemployed adult student" problem [Reiter and Criscuolo, 1981, Poole, 1985], which might be represented as

$$\mathcal{F} = \{\},$$
$$\mathcal{D} = \{student \rightarrow \neg employed,\ student \rightarrow adult,\ adult \rightarrow employed\}.$$

Those writers conclude that adult students are unemployed.)

But then "specificity" is defined in terms of defaults and just means partial membership. But this creates a semantic problem: how can we verify whether *emu* is "more specific" than *bird*?

There seems to be no single solution to the multiple extension problem within that framework. Hanks and McDermott [1986] illustrate "minimizing abnormality" doesn't rid us of "unintended interpretations": it only makes them harder to find. We argue that the methods of "cancellation laws" [McCarthy, 1986] or "constraints" [Poole, 1988], that is, explictly pruning the proof tree, are not much different from explicitly listing all exceptions and considerably less perspicacious. This also alters a probabilistic quality of the defaults: adding "constraints" makes defaults mutually exclusive.

What works well is an idea from probability: conditioning on all observations. This appears to be the direction taken by Geffner [1988] and Delgrande [1988]. We will discuss this later.

2.2 The lottery paradox

Informally stated, the lottery paradox means we generate an inconsistency if we choose to believe all sentences with probability greater than some threshold. As shown by Poole [1989], a qualitative version of this paradox arises if we allow default "proof by contradiction".

For suppose we argue thus. Tweety is a bird. Then Tweety is not an emu. For, suppose to the contrary that Tweety were an emu. Then Tweety, by default, wouldn't fly. But since Tweety is a bird, Tweety, by default, does fly, a contradiction.[1]

As Poole [1989] argues, in many domains it may be that every subclass is unusual in some respect and as a consequence we can show that, by default, a bird is not a member of any subclass of bird, a contradiction.

Even if this possibility does not arise, there are problems. *Emus* indeed may be rare, but in [Neufeld, 1989a], a simple numerical example is constructed where most birds fly and most emus don't yet most birds are emus. As illustrated there and elsewhere, it is also true that non-birds are non-emus, and therefore nothing is an emu. This can lead to an unwanted side effect called the "dingo" paradox [Poole, 1987], where we assume, by default, that an arbitrary individual must belong to a class with no known exceptions . See [Neufeld and Poole, 1988, Neufeld, 1989a] and especially [Kyburg, 1971, Kyburg, 1988, Poole, 1989] for

[1] Technically, the conclusion follows by using both defaults, which are not inconsistent, in a resolution proof.

fuller discussion on lottery paradoxes and default logic.

2.3 Simpson's paradox

Simpson's paradox is usually presented as a problem in sampling; we have in mind the presentation by Blyth [1973] where it is stated as a result in deductive probability theory. The problem the default logic community must address is when to restrict default "proof by cases". That is, we must be able to handle the "interacting defaults" [Reiter and Criscuolo, 1981]

$$a \vee b \rightarrow c,$$
$$a \rightarrow \neg c,$$
$$b \rightarrow \neg c,$$

so as to draw a different conclusion from $a \vee b$ than we would from either disjunct separately. As shown in some detail by Neufeld and Horton [1989], various default logics handle this differently.

Some argue this is nit-picking, but Neufeld and Horton [1989] give the following example. Consider a science faculty where every class contains a large core of the same science students and a handful of different arts students. If there are enough classes, there may be more arts students than science students enrolled in the faculty; by increasing the the number of classes, the proportion of arts students in the disjunction may be made arbitrarily close to one.

If we only know that a student is in *some* class, it is easy to construct a default "proof by cases" that by default she is a science student. But this conflicts with more general knowledge that a random student from the disjunction of classes is an arts student. A formalism for this type of knowledge must not lose information about how the student was selected: a disjunction of uncertainties is different from the uncertainty of a disjunction!

2.4 What causes these problems?

Of course, all of these problems are closely related. All contain naive "chaining" or naive use of contrapositives. All result from ignoring two basic ideas from probablity theory:

1. propositions (or events) *condition each other*. For example, if all emus are birds, the probability that a random individual is an emu actually increases after we learn it is a bird.
2. We can combine inferences, but we cannot combine them as we please. Unadorned default inference lets us combine defaults constrained only by *consistency*; probabilistic inferences can be combined, but constrained by knowledge about *independence*.

3 A probabilistic approach to commonsense reasoning

Rather than retract defeasible propositions on receipt of new evidence, we consider a nonnumeric probabilistic account of commonsense knowledge, where we just increase or decrease our belief in propositions as new knowledge is received. We represent basic beliefs graphically as follows. Nodes which encode propositions (or binary random variables) are connected by single or double arcs, with or without a cross. $p(a|b)$ denotes the conditional probability that a is true given that b is true. The arcs have the following meanings:

$a \rightarrow b$ means $p(b|a) > p(b)$,
$a \not\rightarrow b$ means $p(\neg b|a) > p(\neg b)$,
$a \Rightarrow b$ means $1 = p(b|a) > p(b)$, and
$a \not\Rightarrow b$ means $1 = p(\neg b|a) > p(\neg b)$.

Following [Chung, 1942], if a and b are such that $p(b|a) > p(b)$, then we say that a *favours* b. (In some other writings, we have said that a *confirms* b. Some have suggested that we say a *supports* b or a *is relevant to* b.)

We consider only acyclic graphs: this corresponds to the idea that a proposition cannot influence itself.

We say a is *unconditionally independent* of b if $p(a|b) = p(a)$ whenever the conditional probability is defined. We also say a is *conditionally independent* of b given c if $p(a|bc) = p(a|c) = p(a|\neg bc)$ and $p(a|b\neg c) = p(a|\neg c) = p(a|\neg b\neg c)$ whenever the conditional probabilities are defined.

For simplicity, we define these relations on propositions: they extend in the obvious way for random variables with more than two outcomes [Wellman, 1988].

Lastly, the topology of the graph encodes independence knowledge in the following way. We assume that the joint probability distribution of all propositions in the graph can be factored into the product of the conditional probability distributions of each node given its parents. This is the assumption behind what are called Bayes' nets, [Pearl,1986], influence diagrams [Shachter, 1986], causal networks [Lauritzen and Speigelhalter, 1988]. This means that a proposition is *conditionally independent* of its predecessors and siblings whenever the outcomes of all of its *immediate* predecessors are known. In [Neufeld and Poole, 1988] this is called an *inference graph*; it is based on the same idea as Wellman's [1986, 1987] qualitative influence diagrams.

This assumption means we can quickly detect conditional independence. Geiger and Pearl [1988] show that detecting conditional independence is equivalent to detecting a property on influence diagrams called *d-separability*, and furthermore, there are linear algorithms [Geiger, 1988] for determining this.

From the definition of conditional independence and the axioms of probability theory [Cox, 1946], it is possible to derive the following inference rules [Neufeld, 1989b]:

R0 *(Contraposition)* If $p(a|b) > p(a)$, then $p(\neg b|\neg a) > p(\neg b)$.

R1 *(Symmetry)* If $p(a|b) > p(a)$, then $p(b|a) > p(b)$.

R2 *(Resolution)* If $p(a|b) > p(a)$ and $p(b|c) > p(b)$, and a is conditionally independent of c given b, then $p(a|c) > p(a)$.

R0 shows that we may use "contrapositives" of links, but **R2** states that we cannot chain arbitrarily on links and contrapositives of links.

R1 is, in our opinion, the most important rule in terms of offering insight, if we agree that it is reasonable to draw approximate conclusions on the basis of probabilistic associations. This provides an important connection between default inference and diagnosis: in this system, they are exactly the same activity. But most importantly, this is what gets us out of the lottery paradox.

If we combine **R0** and **R1**, we can show that if $p(a|b) > p(a)$, then $p(\neg a|\neg b) > p(\neg a)$. But there are many more interesting derived inference rules not relevant to the example presented here[Neufeld, 1989b].

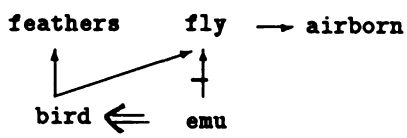


Figure 1: Birds fly graph.

For completeness, we present a familiar example. Figure 1 encodes some "commonsense" knowledge about the "birds fly" domain as an inference graph. For brevity, we write "birds fly" when we mean "*bird* favours *fly*". Thus, birds fly and have feathers, flying things are airborn and emus are birds and don't fly. We can make the following inferences.

1. *Birds fly, emus don't.* Here knowledge about dependence (or lack of conditional independence) does not allow transitive inference from *emu* to *bird* to *fly*, since *fly* is not conditionally independent of *emu*, given *bird*. Thus, we don't have a "multiple extension" problem.
2. *Birds are airborn.* Here the conditional independence knowledge allows the desired transitive inference from *bird* to *fly* to *airborn* since *airborn* is conditionally independent of *bird* given *fly*. Thus some "chaining" is allowed.
3. *Emus do not vanish.* We thus avoid the lottery paradox and its derivatives. Rather than believing a *bird* is not a member of some subclass, we increase belief that a *bird* is a member of some subclass. Thus, observing *bird* increases belief in *emu*. This does not mean that birds, by default, are emus: it means that once we learn something is a bird, we increase our belief that it is some particular kind of bird.
4. *Non-flying things are not birds* Some writers [Geffner, 1988, Poole, 1988] seem to feel that certain unwanted inferences in default logic could be eliminated by restricting use of the contrapositive, but this inference seems natural to us. We believe the problem is with the way sentences are combined.
5. *Feathered things fly* Thus, this system easily allows the kinds of inferences suggested in [Pearl, 1987], that is, reasoning from effects to effects if they have a common cause. The association is natural, but we know of no system other than Pearl's [1987] $C - E$ system that allows it.
6. *Emus are not airborn.* Combining rules **R0** and **R1** lets us "chain" from *emu* to $\neg$*fly* to $\neg$*airborn*. Once again, we know of no other system that allows this inference.

Lastly, for the "arts and science students" example, this system does not in general favour a conclusion given a disjunction of propositions when the conclusion is favoured by every disjunct. Under certain conditions, this inference may be permitted, see [Neufeld and Horton, 1989] for an example.

To sum up, the system permits interesting forward and backward inferences. The conditional independence knowledge encoded in the topology of the graph provides an elegant mechanism for chaining but also "blocks" unwanted inferences.[2] Not only do the emus not fly, we show elsewhere [Neufeld, 1989b] that we can add *flemus* (rare flying emus) to the graph and still get the correct inferences. In that same work, we present the Nixon diamond and the "African and Royal elephants" examples from [Sandewall, 1986, Horty *et al.*, 1987]. While our answers may not agree with the answers expected by other researchers, our formalism provides a way of showing why the "expected" answer is wrong in a statistical sense. (Of course that applies strictly to this framework; the other researchers do not intend our semantics.)

We used to suggest that this probabilistic formalism seemed to be an interesting partial account of defaults, whatever else a default may mean. But we are representing and reasoning about *shifts* in belief, not "jumping" to conclusions. It is surprising nonetheless that these simple semantics appear to produce the expected answer when applied to many of the graphical examples which appear in the default reasoning literature.

[2] In fact, statistical independence is an obvious partial solution to the "frame" problem.

4 Defaults and diagnosis

At least two writers in the default camp have tried to formalize a relationship between default reasoning and diagnosis. The diagnosis problem adds an interesting twist to the multiple extension problem. When doing diagnosis, the idea that the same symptoms have several mutually inconsistent diagnoses is not disagreeable, but most writers (and readers) are uncomfortable with the conclusion that emus, by default, fly.

Providing a uniform treatment of default inference and diagnosis appears to sacrifice the semantics of one for the other. Poole *et al.* [1987] treat default reasoning as a special case of diagnosis; Reiter [1987] treats diagnosis as a special case of default inference. The former approach sacrifices the notion of defaults as statements of typicality but yields an elegant definition of diagnosis as *explanation*. The latter approach retains the notion of defaults as statements of typicality but gives, in our view, a less natural definition of diagnosis.

More precisely, in an early paper, Poole *et al.* [1987] put default inference and diagnosis under the umbrella of conjectural reasoning and are very specific about not treating defaults as statements of typicality or high probability. If we have

$$\mathcal{F} = \{bird,\ cold \rightarrow sneeze\},$$
$$\mathcal{D} = \{cold,\ bird \rightarrow fly\},$$

then *cold* is a conjecture that explains *sneeze* and *bird* $\rightarrow$ *fly* is a conjecture that explains *fly*. On the face of it, both inferences seem reasonable. Surely we don't want to believe that people typically have colds, yet *cold* seems like an obvious explanation for *sneeze* and diagnosis as *explanation* is an attractive idea. But then for the "birds fly" graph of Figure 1, "emus fly" is a legitimate conclusion that must be handled with a separate theory of "explanation comparators" [Poole, 1985]. In fact, in later work, Poole [1987b] formalizes the separation of default and diagnositc inference with the introduction of a new kind of default called a *conjecture*.

Reiter [1987] views diagnosis as maximizing normality given observations of abnormal behaviour (symptons). The defaults are statements of normality and the facts are descriptions of the system under normal and abnormal conditions. Diagnoses are then the smallest set c of normality conditions such that the assumption that every element of c is false together with the assumption all other normality conditions are true is consistent with the facts and observations. While defaults remain statements of typicality, the definition of diagnosis loses the familiar intuition of reasoning backward towards causes.

In this probabilistic framework, default inference is obtained from diagnosis (and vice versa) simply by exchanging the quantities on either side of the conditioning bar: they are truly inverses of one another. This is reasonable since the likelihood ratios are the same regardless of the direction of reasoning:

$$\frac{p(s|d)}{p(s)} = \frac{p(d|s)}{p(d)}.$$

(Suppose that d is a disease and s a symptom and an arc is directed from d into s.) Yet diagnostic inference has qualities very different from default inference because of the nature of the conditional independence knowledge encoded in the graph: we do not get one from the other by reversing the direction of all the arcs in the graph.

The probabilistic framework offers at least three advantages over default-based frameworks.

4.1 Diagnostic heirarchies with exceptions

Relabel the nodes of the "birds fly" graph as in Figure 2.

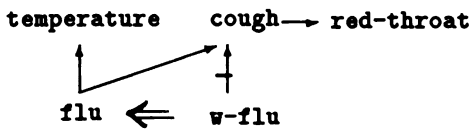


Figure 2: Diagnostic graph with exceptions.

If we observe *cough* and *temperature*, default-based diagnosis runs into a multiple extension problem. In a probabilistic framework, only *flu* is favoured by these observations. If we observe $\neg$*cough* and *temperature*, only *w-flu* is favoured, even though *flu* is a consequence of *w-flu*.

4.2 A characterization of simplicity?

Consider Figure 3. It contains a small graph of the type that appears frequently in the diagnostic literature [Peng and Reggia, 1986], namely, a directed bipartite graph with arcs directed from disorders to symptoms. Suppose we interpret it as an inference graph.

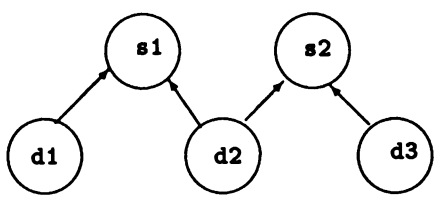


Figure 3: A simple diagnostic network.

Now suppose we observe $s_1 s_2$. Intuition favours d_2 as an explanation and indeed this is the only diagnosis favoured by the observations. Interestingly, $s_1 s_2$

favours d_1 and d_3 neither separately nor together (although s_1 and s_2 separately favour $d_1 d_3$).

This may shed some light on why we prefer the "syntactically" simpler explanation d_2: favouring *or* disfavouring the conjunction simply requires probabilistic information not directly available from the graph.

Note that if the graph is interpreted in the system of Poole et al. [1987], for example, with d_i elements of $\mathcal{D}$ and all links in $\mathcal{F}$, d_2 and $d_1 d_3$ are both minimal diagnoses.

4.3 Defaults and set covering models

The approach of Poole et al. described above yields an elegant semantics for diagnosis as explanation when disorders logically imply symptoms. In fact, the favourability relation automatically holds between disorders and symptoms when this is the case. But to probabilistically justify diagnoses produced by default reasoners or set covering algorithms [Peng and Reggia, 1986] when links are not logical, it is necessary to exploit other knowledge, for example, the "noisy or" model [Wellman, 1988].

Obviously, some models produce answers inconsistent with others: the "noisy or" model cannot be used in domains where conjunctions exhibit unusual behaviour. We are not constrained to interpret graphs according to a uniform semantics: we remark, however, that some attempts to treat models like default assumptions result in the "multiple extension problem" reincarnated in a probabilistic setting.

5 Coherent models

It may be that an intelligent system only answers yes/no queries, for example, "Does the evidence suggest a thyroid condition?", "Do birds fly?". Some systems, for example, robots, are required to construct a provisional model of the world based on very few observations. This raises the interesting question of how to *combine* propositions favoured by a small number of observations.

This problem was inspired by the notion of a *default extension* [Reiter, 1980] which can be defined as the deductive closure of a maximal consistent set of facts and defaults [Poole, 1988]. The original intuition seemed to be that an extension corresponded to an "acceptable set of beliefs" [Reiter, 1980] that could be discarded on learning new knowledge to the contrary. But we feel some default extensions are not acceptable. If we interpret Figure 1, for example, as facts and defaults in the systems of Poole or Reiter, there is an extension where the emu is flying and airborn, because it is a typical bird! Logical consistency seems insufficient for constructing acceptable models. While we don't want a solution that does makes the unlikely impossible, we don't want a system that makes the unlikely acceptable.

Default extensions exhibit other peculiar behaviours. Observe that *bird* has a single extension, whereas *emu* has two. More specific knowledge seems to *increase* rather than decrease confusion! We thus turn to probabilistic justifications for combining beliefs as a way of constructing acceptable models.

One possibility might be the "longest" conjunction of beliefs favoured by the observations. This has a problem: suppose there is a green (g) flying (f) emu (e), and emus are always birds (b). Then we have $1 = p(b|gfe) > p(b)$. But, by the symmetry rule **R1**, then $p(gfe|b) > p(gfe)$. If we observe a green flying emu, it is desirable to believe that we are observing a bird, but if we only observe a bird, it doesn't seem desirable to consider *green flying emu* as an acceptable model. The same probabilistic properties that factor out irrelevant knowledge in some settings make irrelevant predictions in others.

Another candidate might be maximal sets of propositions that *pairwise* favour each other. But then we cannot combine *emu* with $\neg$*fly* and *feathers* since the latter pair do not favour each other.

We settle for the following notion. A set of literals S is *coherent* if for any e in S, $p(e|S-e) > p(e)$. That is, the probability of any sentence in S is increased as a consequence of knowing the other sentences. Then, a *probabilistic extension* of s is a maximal coherent set of sentences S containing s. In the "birds fly" example, the number of *probabilistic* extensions *decreases* as more knowledge is received. Thus *emu* has one extension:

$$\{emu, bird, \neg fly, feathers, \neg airborn\},$$

and *bird* also has

$$\{bird, feathers, fly, airborn\}.$$

No extension contains both *emu* and *fly*.

We explore some interesting uses of probabilistic extensions.

5.1 Diagnosis

For simple graphs like Figure 3, diagnoses are easy to construct. Where there are many arcs, it may be that no disorder or combination of disorders is favoured by the symptoms. (Consider an example pointed out to us by Paul van Arragon: a graph with two disorders and two symptoms and arcs directed from both disorders to both symptoms.) An alternative to the approach of considering other probabilistic models might be the following. If a set of observations is not coherent, break the set into maximal coherent subsets and recursively find diagnoses of the smaller sets; then take the cross product.

5.2 Plan recognition

Kautz [1987] in his thesis investigates *plan recognition*. That is, given an observation, infer the entire set of

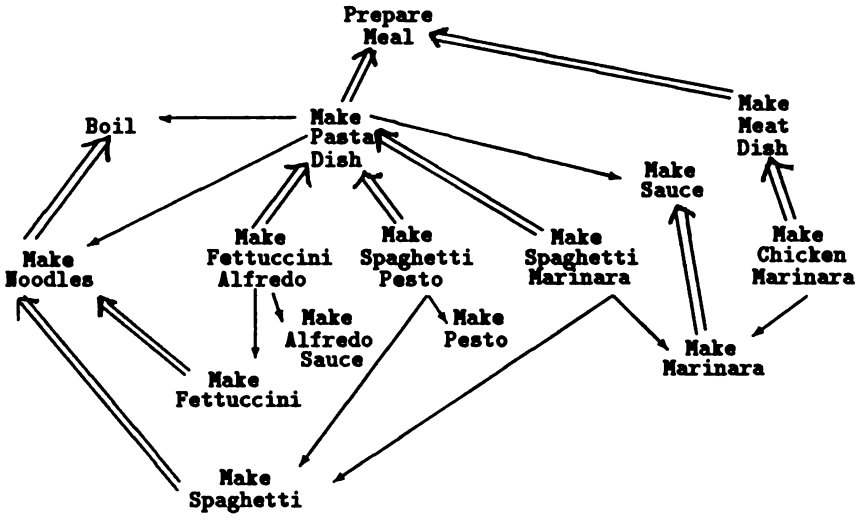


Figure 4: Kautz's presentation of the cooking world.

activities or *plan* surrounding the observation. Probabilistic extensions seem to be have some interesting analogies with plans.

Figure 4 illustrates some aspects of Kautz's "cooking world". We have factored out temporal information.

In Kautz's diagram, double arrows define an *abstraction hierarchy* and single arrows define a *decomposition* hierarchy. The intuition is that we both generalize and specialize when making plans. Kautz's goal is inferring plans from observations. For example, if an agent observes spaghetti or fettuccini being made, she should be able to infer that boiling will occur and be able to help or hinder as suits her.

The domain is not complicated. But if we attempt to "add semantics" of favourability to Figure 4, some of the inferences we ought to make are blocked because of what appears as too many dependencies among many events. For example, it is impossible to favour *make-sauce* given *make-noodles* even though every noodle dish has a sauce.

Therefore, before discussing the relationship between plan recognition and probabilistic extensions, we will first build an inference graph in this domain from first principles. To begin, we notice that there are only a handful of random variables in this domain. Each set of abstraction arrows can be viewed as being directed from a set of outcomes of a particular random variable. For example, there is a random variable *Kind-Of-Meal* with four outcomes, a random variable *Kind-Of-Sauce* with three outcomes, a random variable *Kind-Of-Noodle* with two outcomes, and so on. We present Figure 5 as a suitable basis for Kautz's cooking world.

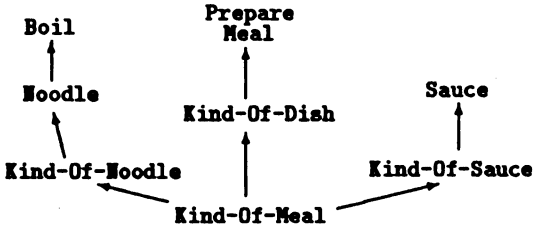


Figure 5: The variables in Kautz's domain.

Note there is no direct arc from *Kind-Of-Dish* to *Boil* or *Sauce*, as would be the case if we interpreted Kautz's diagram as an inference graph. We argue that *Sauce* is directly affected by *Kind-Of-Sauce* and only indirectly by *Kind-of-Meal* and *Kind-Of-Dish*. We find that no inferences are sacrificed, because of the properties of the probabilistic resolution rule. Similarly, *Boil* is directly affected by *Noodles* and not *Kind-Of-Dish*. We complete this graph by adding arcs from the various outcomes of the random variables to get Figure 6.

This graph has a much simpler underlying structure (Figure 5) than apparent from Figure 6 because most of the arcs between propositions are redundant with respect to variables.

Using only rules **R0** to **R2**, it is possible to make many interesting inferences. If we observe *spaghetti*, we favour *boil*. Likewise, if we observe *fettuccini*. If we observe either *spaghetti* or *fettuccini*, we also favour *sauce*, and we favour the appropriate sauce for both.

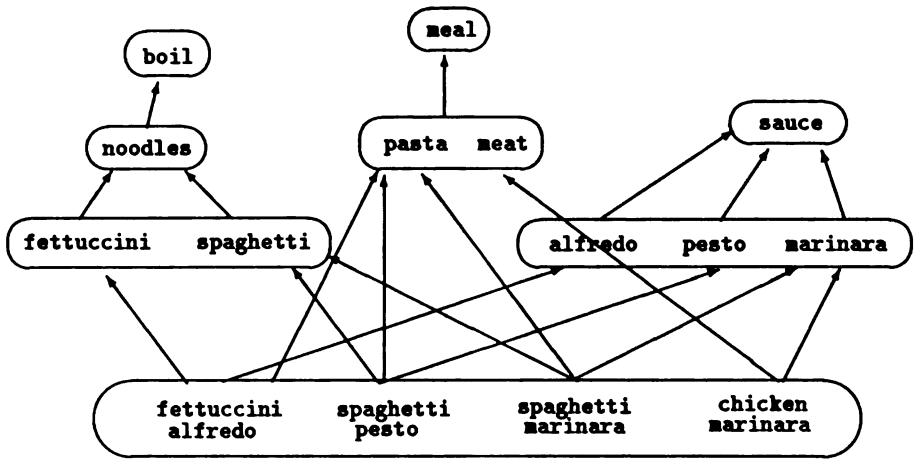


Figure 6: Kautz's cooking world, the inference graph.

All these inferences are made by repeated application of the probabilistic resolution rule **R2**.

We now consider construction of maximal extensions. Straightforward code for computing probabilistic extensions appears in [Neufeld, 1989a]. It appears that computing maximal extensions of any observed event results in plan recognition. For example, if we observe *noodles*, the relevant set of maximal probabilistic extensions is

{*noodles, meal, pasta, spaghetti, pesto, sauce, boil, spaghetti-pesto*},

{*noodles, meal, pasta, spaghetti, marinara, sauce, boil, spaghetti-marinara*},

{*noodles, meal, pasta, fettuccini, alfredo, sauce, boil, fettuccini-alfredo*}.

Suppose we observe *noodles* $\wedge$ *marinara*. There is exactly one extension:

{*noodles, meal, pasta, spaghetti, marinara, sauce, boil, spaghetti-marinara*}.

We might assume the conjuncts are related since they are coherent. In this case, the single extension may be viewed as the single unordered plan that "explains" the observations.

On the other hand, suppose we *know* the cook is simultaneously preparing two different dishes. Then *marinara* has two extensions and *noodles* has three. We might *suspect* the cook is preparing two different dishes if we observe both *fettuccini* and *marinara*, since we can detect that these two observations do not have the extension property. Therefore, we can generate separate extensions for each, and propose that they are parts of separate plans.

There seems to be an interesting relationship between probabilistic extensions and unordered plan recognition.

5.3 User modelling

An idea suggested by Paul Van Arragon is that the method of computing maximal extensions can be applied to the task of user modelling. We follow the same methodology as for plan recognition, but nodes now encode personality traits instead of actions. Given only a single observation, we expect that maximal extensions will represent many "typical" user models. As more observations are made, the number will decrease, just as the number of maximal extensions decreases in the "birds fly" domain on receipt of more specific observations and the number of plans recognized in the cooking world decreases on receipt of observations that are correlated.

6 Is probability inevitable?

A theorem of Cox [1946] gives strong theoretical evidence that standard probability theory lurks behind every numeric formalism for reasoning about uncertainty. The generalization of that work by Aleliunas [Aleliunas,1988] suggests that the same is true for non-numeric formalisms. Both results suggest that formalisms claiming to be new ways about reasoning under uncertainty risk either isomorphism (to probability) or inconsistency. This is certainly food for thought in light of the apparent paradoxes described earlier.

If we interpret the results of Cox and Aleliunas as suggesting the inevitability of probability as the science of sound approximate reasoning, we obtain an

explanation of why default logics seem to be *converging towards probability*.

For example, we stated earlier that many newer default logics use the idea of *conditioning on all observations*. Reiter's [1980] original formalism contains only facts and defaults. Facts include observations (Tweety is a bird) as well as "eternal truths" (emus are birds). There appears to be a need to separate *observations* from *knowledge*. For example, Poole [1985] distinguishes between the facts $\mathcal{F}$ and *contingent* facts in order to choose among competing extensions; more recent formalisms are very explicit. Delgrande [1988] defines a *default theory* as a pair $< D, C >$ where D is a set of necessary and default (conditional) sentences and C is a set of *contingent sentences*. Similarly, Geffner [1988] defines a pair $< K, E >$ where K is a *background context* and E an *evidential context*.

Indeed defaults or conditional sentences are grouped with facts in the system of Delgrande [1988], for example, who argues that default knowledge must be consistent with factual knowledge. That is, it is not permissable to believe the *fact* "emus don't fly", and the *default* "emus do fly". [3]

But then there is a striking similarity between making an inference about the plausibility of *fly* given a context *bird* in light of a body of knowledge D (which may contain uncertain knowledge), and making an inference about the posterior of *fly* given the observation *bird* and a body of knowledge (an inference graph, for example, or any other body containing certain and uncertain knowledge.)

Lastly, several default systems are introducing ideas similar to *independence*. Delgrande [1988] uses an *assumption of relevance* and Geffner [1988] uses an idea similar to graph separability to define a notion of *irrelevance*. Pearl [1987] describes the $C - E$ system which appears to encode a test for d-separability in the inference rules.

7 Other probabilistic approaches

Many others are using probability as a replacement or semantics for default inference in other ways. For example, Geffner [1988] explicitly draws on probability for default inference rules. In [Pearl, 1987b], Pearl describes his idea of ϵ-semantics for inheritance hierarchies with exceptions, treating defaults as conditional probabilities close to unity. Bacchus [1988] considers the inferences possible when links are viewed as conditional probabilities greater than c, a domain constant greater than one-half. Buntine [1988], in a recent work, presents a qualitative probabilistic system inspired by Delgrande's [1988] work.

[3] Some systems [Reiter, 1980, Poole *et al.*, 1987] ignore contradictory defaults. Poole *et al.* suggest that we may be prepared to assume a sentence or its contrary to explain some observation.

We prefer to think of this as the study of nonnumeric probabilistic inference; we observe that these formalisms all offer *testable* accounts of commonsense reasoning.

8 Verifiability of commonsense reasoning formalisms

We present inference graphs as mathematical objects. While the performance of the system seems reasonable for the examples in the nonmonotonic literature, we are cautious about entering sensitive problem domains. This system reasons about *shifts* in belief, but a belief may shift upwards yet remain very small. For instance, a pregnant woman may avoid an X-ray even though the probability of damaging the fetus is small in absolute terms. This is generally viewed as sensible. However, the argument that one should buy lottery tickets because the probability of becoming a millionaire is just as slightly increased, is not as generally viewed as sensible.

The problem, therefore, of mathematically deciding which sentences of favourability (even assuming no measurement errors) should appear on an inference graph we presently consider to be very difficult. (That doesn't prohibit anyone from constructing an inference graph where all nodes and arcs encode knowledge meaningful to that individual.)

What appears less difficult is *verification* of inference graphs once they have been constructed. In a word, there are many ways of constructing experiments that to verify whether the knowledge encoded in an inference graph is true in the problem domain. The statisticians have many suggestions here.

All the probabilistic systems described in the previous section are testable in this sense and so are all systems based strictly on first order logic. We therefore propose that *verifiability* be a requirement of *any* formalism that purports to reason in the so-called commonsense domain. Although certain "trademark" [Buntine, 1988] properties of default reasoning can be drawn from the literature, to our knowledge, no one in the camp of default logic has proposed any means of testing the correctness of a representation of a particular problem. We believe that the pressing problem in knowledge representation is not in deciding upon the *formalism* or *language*, but deciding on how to go from the problem domain to the formalism and vice versa.

We suggest that a suitable test or experiment would be described by providing formal definitions for all constructs such as defaults, abnormality predicates, conjectures, communication conventions and so on. With such definitions, proofs of equivalence, where they exist, between formalisms would proceed quickly. Moreover, the meaning of differences between formalisms would also be discovered quickly. The initial work on probabilistic treatments of defaults suggests that

some differences will be explained in terms of subtle independence assumptions or thresholds.

We believe this suggestion to be timely; besides the various logic-based approaches cited, there are a variety of graph-based reasoning formalisms [Sandewell, 1986, Horty *et al.*, 1987] addressing the same problems and from within that camp there also seems to be disagreement on what answers these formalisms *ought* to produce. If the experiments we suggest cannot be defined, we do not see how even the most trivial aspects of the debate can be resolved.

9 Conclusions

The most surprising result of this work has been its simplicity. From just the axioms of probability and the ideas of conditional independence and favourability, we produce a compact knowledge representation formalism that generates the expected answer for most problems we have encountered in the default literature. Furthermore, it does not require numerical probabilities, a long standing argument against the use of probability in AI [McCarthy and Hayes, 1969]. Lastly, the notion of coherent probabilistic extensions seems to provide a useful tool for constructing acceptable models of the world.

In the discussion of [Lauritzen and Speigelhalter, 1988], Pearl quotes Glen Shafer as saying "probability is not really about numbers, it is about the structure of reasoning." We believe that this approach, along with other nonnumeric probabilistic approaches, supports this view of probability as a theory of approximate argument.

We close by repeating that our formalism has the important advantage over default logics that we can *test* the correctness of a representation by performing a statistical experiment. To the best of our knowledge, no one in the default logic camp has described an equivalent experiment; we believe that it is timely that such experiments be defined.

Acknowledgements

This work is based mainly on my PhD. thesis which was supervised by Romas Aleliunas and David Poole. The work of [Koopman, 1940] convinced me that some interesting nonnumeric probabilistic inference was possible. Romas Aleliunas provided a great deal of encouragement for this direction; David Poole kept me up to date on the state of default logics and made available an implementation for comparison. Both made many valuable suggestions.

Thanks to colleagues at the University of Waterloo and the University of New Brunswick for critical comments on versions of this paper, including André Trudel and Joe Horton. Thanks to Steven Osbourne and Mike MacDonald for typesetting support at UNB.

References

[Aleliunas, 1988] Romas Aleliunas. A new normative theory of probabilistic logic. In *Proceedings of the Seventh Biennial Conference of the CSCSI*, pages 67–74, 1988.

[Bacchus, 1988] Fahiem Bacchus. A heterogeneous inheritance system based on probabilities. Technical Report 87-03, University of Alberta Centre for Machine Intelligence, 1987.

[Blyth, 1973] Colin Blyth. Simpson's paradox and mutually favorable events. *Journal of the American Statistical Association* 68(343):746, 1973.

[Buntine, 1988] Wray Buntine. Default and likelihood reasoning ≈ qualitative probabilistic reasoning. Submitted.

[Chung, 1942] K-L. Chung. On mutually favorable events. *Annals of Mathematical Statistics*, 13:338–349, 1942.

[Cox, 1946] Richard T. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 14(1):1–13, 1946.

[Delgrande, 1988] James P. Delgrande. An approach to default reasoning based on a first-order conditional logic: revised and extended report. *Artificial Intelligence*, 36:63–90, 1988.

[Etherington, 1988] David W. Etherington. Nonmonotonic reasoning: is the answer harder than the question? Invited talk, *Seventh Biennial Conference of the CSCSI*, (1988).

[Etherington and Reiter, 1983] David W. Etherington and Raymond Reiter. On inheritance hierarchies with exceptions. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 104–108, 1983.

[Geffner, 1988] Hector Geffner. A logic for defaults. In *Proceedings AAAI-88*, pages 449-454, 1988.

[Geffner and Pearl, 1987] Hector Geffner and Judea Pearl. A framework for reasoning with defaults. Technical Report TR-94b, UCLA Computer Science Department, August 1987.

[Geiger and Pearl, 1988] Dan Geiger and Judea Pearl. Logical and algorithmic properties of conditional independence. Technical Report CSD 870056, UCLA Computer Science Department, March 1988.

[Geiger, 1988] Dan Geiger. PhD. thesis, in progress.

[Grosof, 1985] Benjamin Grosof. Default reasoning as circumscription. In *Proceedings of the AAAI Workshop on Nonmonotonic Logic*, pages 115–124, (1985)

[Hanks and McDermott, 1986] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logic

and the frame problem. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 328–333, 1986.

[Horty *et al.*, 1987] J.F. Horty, David S. Touretzky, and Richmond H. Thomason. A clash of intuitions: the current state of nomonotonic multiple inheritance systems. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 476–482, 1987.

[Kautz, 1987] H. Kautz. *A formal theory of plan recognition*. PhD thesis, University of Rochester, 1987.

[Konolige, 1987] Kurt Konolige. On the relation between default and autoepistemic logic. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages xxx–xxx, (1987).

[Koopman, 1940] B.O. Koopman. The axioms and algebra of intuitive probability. *Annals of Mathematics*, 42:269–292, 1940.

[Kyburg, 1971] Henry E. Kyburg, Jr. *Logical Foundations of Statistical Inference*. Kluwer Academic, Dordrecht, Holland, 1971.

[Kyburg, 1988] Henry E. Kyburg, Jr. Probabilistic inference and non-monotonic inference. In *Proceedings 4th AAAI Workshop on Uncertainty*, pages 229–236, 1988.

[Lauritzen and Speigelhalter, 1988] S.L. Lauritzen and D.J. Speigelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society B*, pages 157–224, 1988.

[McCarthy, 1986] J. McCarthy. Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence*, 28:89–118, 1986.

[McCarthy and Hayes, 1969] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In D. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[McDermott and Doyle, 1980] Drew McDermott and Jon Doyle. Non-monotonic logic 1. *Artificial Intelligence*, 13:41–72, 1980.

[Moore, 1985] Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:272–279, 1985.

[Neufeld, 1989a] Eric Neufeld. Choosing reference classes and building provisional models. Submitted.

[Neufeld, 1989b] Eric Neufeld. A probabilistic commonsense reasoner. Submitted.

[Neufeld and Horton, 1989] Eric Neufeld and J.D. Horton. Conditioning on disjunctive knowledge: defaults and probabilities. Submitted to *IJCAI-89*.

[Neufeld and Poole, 1988] Eric Neufeld and David Poole. Probabilistic semantics and defaults. In *Proceedings 4th AAAI Workshop on Uncertainty*, pages 88–99, 1988.

[Pearl, 1986] Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.

[Pearl, 1987a] Judea Pearl. Embracing causality in formal reasoning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 369–373, July 1987.

[Pearl, 1987b] Judea Pearl. Probabilistic semantics for inheritance hierarchies with exceptions. Technical Report CSD870052, UCLA Computer Science Department, 1987.

[Peng and Reggia, 1986] Yun Peng and James Reggia. Plausibility of diagnostic hypotheses: The nature of simplicity. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 140–145, 1986.

[Poole, 1989] David L. Poole. What the lottery paradox tells us about default reasoning. This volume.

[Poole, 1988] David L. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–48, 1988.

[Poole, 1987a] David L. Poole. Fixed predicates in default reasoning. submitted, 1987.

[Poole, 1987b] David L. Poole. Defaults and conjectures: Hypothetical reasoning for explanation and prediction. Technical Report CS-87-54, University of Waterloo Department of Computer Science, 1987.

[Poole, 1985] David L. Poole. On the comparison of theories: Preferring the most specific explanation. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 144–147, 1985.

[Poole *et al.*, 1987] David L. Poole, R.G. Goebel, and R. Aleliunas. Theorist: a logical reasoning system for defaults and diagnosis. In Nick Cercone and Gordon McCalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*. Springer-Verlag, New York, 1987.

[Reiter, 1987] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[Reiter, 1980] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Reiter and Crisculo, 1981] Raymond Reiter and Giovanni Crisculo. On interacting defaults. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 270–276, 1981.

[Sandewell, 1986] Erik Sandewall. Nonmonotonic inference rules for multiple inheritance systems with exceptions. *Proceedings of the IEEE*, 74:1345–1353, October 1986.

[Shachter, 1986] Ross D. Shachter. Evaluating influence diagrams. *Operations Research*, 34, December 1986.

[Wellman, 1988] Michael P. Wellman. *Formulation of Tradeoffs in Planning under Uncertainty.* PhD. Thesis, Massachusetts Institute of Technology, 1988. Available as Technical Report MIT/LCS/TR-427.

[Wellman, 1987] Michael P. Wellman. Probabilistic semantics for qualitative influences. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 660–664, 1987.

[Wellman, 1986] Michael P. Wellman. Qualitative probabilistic networks for planning under uncertainty. In *Proceedings 2nd AAAI Workshop on Uncertainty*, pages 311–318, 1986.

ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus

Edwin P.D. Pednault
AT&T Bell Laboratories
Crawfords Corner Road
Holmdel, New Jersey 07733

Abstract

This paper presents a planning formalism called ADL for representing and reasoning about the effects of actions. ADL integrates the semantics and much of the expressive power of the situation calculus with the notational and computational benefits of the STRIPS language. By combining aspects of both, ADL overcomes the limited expressiveness and semantic difficulties of the STRIPS language without encountering the computational barriers of the situation calculus.

1. Introduction

The tradeoff between the expressive power of a representational language and its computational tractability is well-known in the field of knowledge representation (e.g., Levesque and Brachman 1985). This tradeoff likewise extends to the representations used in automatic planning, as can be seen by comparing the situation calculus (McCarthy and Hayes 1969) with the STRIPS operator language (Fikes and Nilsson 1971).

The situation calculus is highly expressive, enabling one to model quite complex actions. However, the computational costs that accompany it prevent its use in solving all but the simplest of problems (e.g., Green 1969). With the STRIPS language, on the other hand, one can employ an efficient, *vivid* style of reasoning (Levesque 1986) in which theorem-proving is accomplished via database retrievals. The language also has a very intuitive syntax compared to the situation calculus, and it frees one from having to specify frame axioms (c.f., Hayes 1973; Brown 1987). The disadvantage is that only actions with very simple effects can be represented. The STRIPS language also suffers from certain semantic difficulties that lead to unsound inferences in some cases (Lifschitz 1987a).

The obvious solution to the tradeoff issue is to choose a formalism that is just adequate for the problem of interest without being intractable. In view of this approach, we present a hybrid planning formalism called ADL (Action Description Language) that integrates the semantics and much of the expressive power of the situation calculus with the notational and computational benefits of the STRIPS language. By combining aspects of both, ADL overcomes the limited expressiveness and semantic difficulties of the STRIPS language without encountering the computational barriers of the situation calculus.

Syntactically speaking, ADL schemas resemble STRIPS operators augmented with conditional add and delete lists. This syntax enables situation-dependent effects to be described, and yields a level of expressiveness approaching that of the situation calculus. Like STRIPS, ADL also embodies a frame assumption that eliminates the need to specify frame axioms. Thus, one need only describe the changes made when an action is performed, not what remains unaltered.

The semantics of ADL and STRIPS, however, are quite different. Whereas STRIPS operators define transformations on formulas, ADL schemas define transformations on the states themselves (Pednault 1987). The states in this case are the *algebraic structures* (i.e., *Tarskian models*, *interpretations*, etc.) that underlie the semantics of first-order logic (Shoenfield 1967; Van Dalen 1980). In this respect, the semantics of ADL can be directly related to the semantics of first-order *dynamic* logic (Harel 1979; Kautz 1982).

ADL can also be viewed as providing a convenient syntax for a restricted form of the situation calculus. For every ADL schema, there is an equivalent set of axioms in this restricted form, and vice versa. The restricted form is interesting in its own right, since it incorporates a solution to the frame problem that does not rely on circumscription (c.f., McCarthy 1986; Hanks and McDermott 1986, 1987; Kautz 1986; Lifschitz 1986, 1987b; Shoham 1986). The appropriate frame axioms are derived directly from the state-change axioms and reflect the frame assumption embodied in ADL.

In terms of its computational efficiency, ADL lies somewhere between STRIPS and the situation calculus. On one hand, when the initial state is completely known, an efficient, vivid style of reasoning can be employed. This style of reasoning combines the use of regression

operators (Waldinger 1977) with query evaluation techniques from relational database theory (e.g., Ullman 1982; Wiederhold 1983). However, when the initial state is only partially known, a more general form of theorem-proving must be used. This increases the computational complexity. The increase, of course, is inescapable and will necessarily be present in any representational language (e.g., Levesque 1986).

2. Parallels with the Situation Calculus

Let us first consider the semantics of ADL and its relationship to the situation calculus. The situation calculus is a discipline for constructing first-order theories about the effects of actions. To use the calculus, an additional argument is added to every relation and function whose value can change when an action is performed. This additional argument ranges over situations and permits one to reason about the values of relations and functions at different points in time.

Once situational arguments have been added, axioms can be written that describe the effects of actions in different situations. These axioms typically have one of two forms. Axioms of the first kind describe the changes made when an action is performed and have the form

$$\forall s[\pi(s)\wedge\varphi_1(s)\rightarrow\psi_1(result(a,s))].$$

In this schema, s ranges over situations, $\pi(s)$ describes the preconditions for the execution of action a, $\psi_1(result(a,s))$ describes a condition that becomes true as a result of performing a in situation s, and $\varphi_1(s)$ describes the conditions under which this change takes place. There may also be some additional quantified variables other than the situation variable s, depending on the nature of the formulas π, ψ_1, and φ_1, and on whether a is a parameterized action.

Axioms of the second kind describe properties of the world that are not affected by an action. These axioms are called *frame axioms* and have the form

$$\forall s[\pi(s)\wedge\varphi_2(s)\wedge\psi_2(s)\rightarrow\psi_2(result(a,s))]$$

where $\psi_2(result(a,s))$ is obtained by substituting $result(a,s)$ for every free occurrence of s in $\psi_2(s)$. The formula ψ_2 describes a condition that remains unaltered when action a is performed, while φ_2 describes the circumstances under which this occurs. As before, $\pi(s)$ describes the preconditions for the execution of a.

The semantics of ADL can be viewed as a restricted form of the situation calculus in which certain constraints are imposed on the kinds of axioms allowed. The first constraint of this restricted form is that separate axioms must be written for each individual relation and function when describing the effects of an action. Thus, ψ_1 and ψ_2 in the axiom schemas above must be either atomic formulas or their negations. The second constraint assumes that the only effects of an action are those explicitly mentioned in the state-change axioms. This allows the necessary frame axioms to be derived directly from these state-change axioms.

Given the first constraint, the state-change axioms for a relation symbol R will have the following forms:

$$\begin{aligned}&\forall x_1,\ldots,x_n,s[\pi(s)\wedge\alpha_R(x_1,\ldots,x_n,s)\\&\qquad\rightarrow R(x_1,\ldots,x_n,result(a,s))]\\&\forall x_1,\ldots,x_n,s[\pi(s)\wedge\delta_R(x_1,\ldots,x_n,s)\\&\qquad\rightarrow\neg R(x_1,\ldots,x_n,result(a,s))].\end{aligned}\qquad(1)$$

An α_R axiom defines the conditions under which $R(x_1,\ldots,x_n,result(a,s))$ is made true, whereas a δ_R axiom defines the conditions under which $R(x_1,\ldots,x_n,result(a,s))$ is made false. Accordingly, we will refer to $\alpha_R(x_1,\ldots,x_n,s)$ as an *add condition* for symbol R, while $\delta_R(x_1,\ldots,x_n,s)$ will be referred to as a *delete condition*. Note that $\alpha_R(x_1,\ldots,x_n,s)$ and $\delta_R(x_1,\ldots,x_n,s)$ cannot both be true for identical instances of their variables, since the axioms given above would then be inconsistent.

Without loss of generality, we can assume that there is exactly one α_R and one δ_R axiom for each relation symbol R. This can be done, since a collection of axioms of the form

$$\begin{aligned}&\forall x_1,\ldots,x_n,s[\pi(s)\wedge\varphi_1\rightarrow\psi]\\&\qquad\vdots\\&\forall x_1,\ldots,x_n,s[\pi(s)\wedge\varphi_n\rightarrow\psi]\end{aligned}$$

is equivalent to the single axiom

$$\forall x_1,\ldots,x_n,s[\pi(s)\wedge(\varphi_1\vee\ldots\vee\varphi_n)\rightarrow\psi]$$

If action a cannot make $R(x_1,\ldots,x_n,result(a,s))$ true for any value of $x_1,\ldots,x_n,s$, then $\alpha_R(x_1,\ldots,x_n,s)$ must be the formula *FALSE*. Likewise, if $R(x_1,\ldots,x_n,result(a,s))$ cannot be made false, $\delta_R(x_1,\ldots,x_n,s)$ must be *FALSE*.

For function symbols, the state-change axioms have the following form:

$$\begin{aligned}&\forall x_1,\ldots,x_n,y,s[\pi(s)\wedge\mu_F(x_1,\ldots,x_n,y,s)\\&\qquad\rightarrow F(x_1,\ldots,x_n,result(a,s))=y]\end{aligned}\qquad(2)$$

where $\mu_F(x_1,\ldots,x_n,y,s)$ defines the conditions under which $F(x_1,\ldots,x_n,y,result(a,s))$ is assigned the value y. For this reason, $\mu_F(x_1,\ldots,x_n,y,s)$ will be referred to as an *update condition* for symbol F. Here too we can assume without loss of generality that there is exactly one μ_F formula for each function symbol F. If $F(x_1,\ldots,x_n,result(a,s))$ is not assigned a value for any instance of $x_1,\ldots,x_n,s$, then $\mu_F(x_1,\ldots,x_n,y,s)$ must be the formula *FALSE*.

Since functions are single-valued, assigning a value to $F(x_1,\ldots,x_n,result(a,s))$ automatically causes the "old" value to be "deleted". This is expressed by the following theorem, which is derived from Axiom (2):

$$\begin{aligned}&\forall x_1,\ldots,x_n,s[\pi(s)\wedge\exists y(\mu_F(x_1,\ldots,x_n,y,s)\wedge F(x_1,\ldots,x_n,s)\neq y))\\&\qquad\rightarrow F(x_1,\ldots,x_n,result(a,s))\neq F(x_1,\ldots,x_n,s)]\end{aligned}$$

Consequently, there is no need for the equivalent of delete conditions when dealing with functions. Note also that there can be at most one value of y for which $\mu_F(x_1,\ldots,x_n,y,s)$ is true for each instantiation of the variables $x_1,\ldots,x_n,s$, since otherwise we would be attempting to assign $F(x_1,\ldots,x_n,result(a,s))$ more than one value.

Frame axioms are derived from state-change axioms by applying the second constraint of the restricted situation calculus. For a relation symbol R, if $R(x_1,\ldots,x_n,s)$ is true and $R(x_1,\ldots,x_n,result(a,s))$ is not explicitly made false, then we assume that nothing else occurs to make it false. Similarly, if $R(x_1,\ldots,x_n,s)$ is false and $R(x_1,\ldots,x_n,result(a,s))$ is not explicitly made true, we assume that nothing else occurs to make it true. The truth value of $R(x_1,\ldots,x_n,result(a,s))$ will then be the same as $R(x_1,\ldots,x_n,s)$ in these cases. The appropriate frame axioms can then be expressed in terms of the add and delete conditions for R as follows:

$$\forall x_1,\ldots,x_n,s\big[\pi(s)\wedge R(x_1,\ldots,x_n,s)\wedge\neg\delta_R(x_1,\ldots,x_n,s)\rightarrow R(x_1,\ldots,x_n,result(a,s))\big]$$

$$\forall x_1,\ldots,x_n,s\big[\pi(s)\wedge\neg R(x_1,\ldots,x_n,s)\wedge\neg\alpha_R(x_1,\ldots,x_n,s)\rightarrow\neg R(x_1,\ldots,x_n,result(a,s))\big]. \quad (3)$$

Together, the state-change and frame axioms imply that the truth value of $R(x_1,\ldots,x_n,result(a,s))$ is completely determined by the add conditions, the delete conditions, and the truth value of $R(x_1,\ldots,x_n,s)$ when action a is executable.

For function symbols, we assume that the value of $F(x_1,\ldots,x_n,result(a,s))$ will be the same as $F(x_1,\ldots,x_n,s)$ unless it is explicitly updated. The frame axiom for F is then

$$\forall x_1,\ldots,x_n,s\big[\pi(s)\wedge\neg\exists y\big(\mu_F(x_1,\ldots,x_n,y,s)\big)\rightarrow F(x_1,\ldots,x_n,result(a,s))=F(x_1,\ldots,x_n,s)\big] \quad (4)$$

Note that this axiom and the state-change axiom for F together imply that, when action a is executable, the value of $F(x_1,\ldots,x_n,result(a,s))$ is completely determined by the update conditions for F and the value of $F(x_1,\ldots,x_n,s)$.

3. The Syntax of ADL

The syntax of ADL is designed to provide a convenient means of specifying the α, δ, μ, and π formulas that define an action. The syntax resembles that of the STRIPS operator language augmented with conditional add and delete lists. The action schemas shown in Figure 1 illustrate this syntax. The first schema, Put(b,l), models the blocks-world action of stacking one block atop another or placing the block on the table. The second schema, Pour(p,q), is inspired by a mathematical puzzle and models the action of pouring a liquid from one container into another until either the first is emptied or the second is filled to capacity. The third schema, Adjust (a',θ',t'), models the action of adjusting the thrust and attitude of a lunar lander and serves to demonstrate how time and continuous processes might be modeled.

As the examples illustrate, an ADL schema consists of an action name, an optional parameter list, and four optional groups of clauses labeled PRECOND, ADD, DELETE, and UPDATE. The PRECOND group consists of a list of formulas that define the preconditions for the execution of the action. Every formula in the list must be true when the action is performed; hence, the overall precondition π is the conjunction of these formulas. If the list is absent, π is taken to be the formula $TRUE$, meaning that the action may be performed in every state. For instance, the precondition of Pour(p,q) is the formula $TRUE$, whereas for Put(b,l) it is

$$(b\neq l)\wedge(b\neq TABLE)\wedge\forall z\neg On(z,b)\wedge(l=TABLE\vee\forall z\neg On(z,l)).$$

The α and δ formulas are specified by the ADD and DELETE groups, respectively. Each group consists of a set of clauses of the forms shown in the left hand column of Table 1. In this table, R is a relation symbol, $\tau_1,\ldots,\tau_n$ are terms, ψ is a formula, and $z_1,\ldots,z_k$ are variable symbols that appear in terms $\tau_1,\ldots,\tau_n$ but do not appear in the parameter list of the action schema.

Each clause in an ADD group corresponds to an add condition $\hat{\alpha}_R$ for a relation symbol R. These conditions are defined in the right hand column of Table 1. The formula α_R that defines the overall add condition for R is the disjunction of the individual $\hat{\alpha}_R$'s. If no add conditions are specified for R, α_R is taken to be the formula $FALSE$ (i.e., nothing is added to relation R). For example, the overall add condition $\alpha_{Above}(x,y)$ for Put(b,l) is the formula

$$(x=b\wedge y=l)\vee\exists z\,[x=b\wedge y=z\wedge Above(l,z)] \quad (5)$$

The semantics of the DELETE group is similar to the ADD group, except in this case each clause corresponds to a delete condition $\hat{\delta}_R$ (see Table 1). The disjunction of the individual $\hat{\delta}_R$'s defines the overall delete condition δ_R for relation R. If no delete conditions are specified for R, δ_R is taken to be the formula $FALSE$ (i.e., nothing is to be deleted from R). For example, the overall delete condition $\delta_{Above}(x,y)$ for Put(b,l) is the formula

$$\exists z\,[x=b\wedge y=z\wedge z\neq l\wedge\neg Above(l,z)] \quad (6)$$

UPDATE groups are used to specify the μ-formulas for updating the values of function and constant symbols. An UPDATE group consists of a set of clauses of the forms shown in the left hand column of Table 2. In this table, C is a constant symbol, F is a function symbol, $\tau_1,\ldots,\tau_n,\tau$ are terms, ψ is a formula, and $z_1,\ldots,z_k$ are variable symbols that appear in terms $\tau_1,\ldots,\tau_n,\tau$ but not in the parameter list of the action schema. Each clause

Put(b,l) ;;; *Place block b on top of object l*

Precond: $b \neq l$, $b \neq TABLE$, $\forall z \neg On(z,b)$, $(l = TABLE \vee \forall z \neg On(z,l))$

Add: On(b,l)

Above(b,l)

Above(b,z) for all z such that Above(l,z)

Delete: On(b,z) for all z such that $z \neq l$

Above(b,z) for all z such that $z \neq l \wedge \neg$Above(l,z)

Pour(p,q) ;;; *Pour the contents of vessel p into vessel q*

Update: Volume(q) $\leftarrow$ min(Capacity(q), Volume(p) + Volume(q))

Volume(p) $\leftarrow$ max(0, Volume(p) - Capacity(q) + Volume(q))

Adjust(a',θ',t') ;;; *Set acceleration to a' and attitude to θ' at time t'*

Precond: $t_o \leq t'$, $min \leq a' \leq max$

Update: $t_o \leftarrow t'$

$a(t) \leftarrow a'$ for all t such that $t \geq t'$

$\theta(t) \leftarrow \theta'$ for all t such that $t \geq t'$

$v_x(t) \leftarrow a' \sin \theta' \ (t-t') + v_x(t')$ for all t such that $t \geq t'$

$v_y(t) \leftarrow (g - a' \cos \theta')(t - t') + v_y(t')$ for all t such that $t \geq t'$

$x(t) \leftarrow {}^1/_2 a' \sin \theta' \ (t-t')^2 + v_x(t')(t - t') + x(t')$ for all t such that $t \geq t'$

$y(t) \leftarrow {}^1/_2 (g - a' \cos \theta') \ (t-t')^2 + v_y(t')(t - t') + x(t')$ for all t such that $t \geq t'$

Figure 1: Examples of actions formulated in ADL

Clause	$\hat{\alpha}_R(x_1,\ldots,x_n)/\hat{\delta}_R(x_1,\ldots,x_n)$
$R(\tau_1,\ldots,\tau_n)$	$(x_1 = \tau_1 \wedge \cdots \wedge x_n = \tau_n)$
$R(\tau_1,\ldots,\tau_n)$ if ψ	$(x_1 = \tau_1 \wedge \cdots \wedge x_n = \tau_n \wedge \psi)$
$R(\tau_1,\ldots,\tau_n)$ for all $z_1,\ldots,z_k$	$\exists z_1,\ldots,z_k(x_1 = \tau_1 \wedge \cdots \wedge x_n = \tau_n)$
$R(\tau_1,\ldots,\tau_n)$ for all $z_1,\ldots,z_k$ such that ψ	$\exists z_1,\ldots,z_k(x_1 = \tau_1 \wedge \cdots \wedge x_n = \tau_n \wedge \psi)$

Table 1. Add/Delete Clauses and Their Meanings

corresponds to an update condition $\hat{\mu}_F$ for a function symbol F, or an update condition $\hat{\mu}_C$ for a constant symbol C. These conditions are defined in the right hand column of Table 2. The disjunction of the individual update conditions defines the overall update condition μ_F/μ_C for F/C. If no update conditions are specified for F/C, then μ_F/μ_C is taken to be the formula *FALSE* (i.e., no updates take place for that symbol). For example, $\mu_{Volume}(x,y)$ for Pour(p,q) is the formula

$$[x = q \wedge y = \min(\text{Capacity}(q), \text{Volume}(p) + \text{Volume}(q))]$$
$$\vee [x = p \wedge y = \max(0, \text{Volume}(p) - \text{Capacity}(q) + \text{Volume}(q))].$$

As with the STRIPS language, situational arguments are not used in ADL. To translate ADL schemas into equivalent sets of axioms in the situation calculus, it is necessary to introduce situational arguments in the α, δ, μ, and π formulas derived from the schemas. For example, $\alpha_{Above}(x,y)$ for Put(b,l) becomes the formula

Clause	$\hat{\mu}_R(y)/\hat{\mu}_R(x_1,\ldots,x_n,y)$
$C \leftarrow \tau$	$(y=\tau)$
$C \leftarrow \tau$ if ψ	$(y=\tau \wedge \psi)$
$C \leftarrow \tau$ for all $z_1,\ldots,z_k$	$\exists z_1,\ldots,z_k(y=\tau)$
$C \leftarrow \tau$ for all $z_1,\ldots,z_k$ such that ψ	$\exists z_1,\ldots,z_k(y=\tau \wedge \psi)$
$F(\tau_1,\ldots,\tau_n) \leftarrow \tau$	$(x_1=\tau_1 \wedge\cdots\wedge x_n=\tau_n \wedge y=\tau)$
$F(\tau_1,\ldots,\tau_n) \leftarrow \tau$ if ψ	$(x_1=\tau_1 \wedge\cdots\wedge x_n=\tau_n \wedge y=\tau\wedge\psi)$
$F(\tau_1,\ldots,\tau_n) \leftarrow \tau$ for all $z_1,\ldots,z_k$	$\exists z_1,\ldots,z_k(x_1=\tau_1 \wedge\cdots\wedge x_n=\tau_n \wedge y=\tau)$
$F(\tau_1,\ldots,\tau_n) \leftarrow \tau$ for all $z_1,\ldots,z_k$ such that ψ	$\exists z_1,\ldots,z_k(x_1=\tau_1 \wedge\cdots\wedge x_n=\tau_n \wedge y=\tau\wedge\psi)$

Table 2. Update Clauses and Their Meanings

$\alpha_{Above}(x,y,s) \equiv (x=b \wedge y=l) \vee \exists z[x=b \wedge y=z \wedge Above(l,z,s)]$.

Likewise, $\delta_{Above}(x,y)$ and π become

$$\delta_{Above}(x,y,s) \equiv \exists z[x=b \wedge y=z \wedge z\neq l \wedge \neg Above(l,z,s)]$$

$$\pi(s) \equiv (b\neq l) \wedge (b\neq TABLE) \wedge \forall z \neg On(z,b,s) \wedge (l=TABLE \vee \forall z \neg On(z,l,s)) .$$

The effect of Put(b,l) on the Above relation would then be axiomatized in the situation calculus by two state-change axioms and two frame axioms:

$$\forall b,l,x,y,s[(b\neq l) \wedge (b\neq TABLE) \wedge \forall z \neg On(z,b,s)) \wedge (l=TABLE \vee \forall z \neg On(z,l,s)) \wedge [(x=b \wedge y=l) \vee \exists z(x=b \wedge y=z \wedge Above(l,z,s))] \rightarrow Above(x,y,result(Put(b,l),s))]$$

$$\forall b,l,x,y,s[(b\neq l) \wedge (b\neq TABLE) \wedge \forall z \neg On(z,b,s)) \wedge (l=TABLE \vee \forall z \neg On(z,l,s)) \wedge \exists z[x=b \wedge y=z \wedge z\neq l \wedge \neg Above(l,z,s)] \rightarrow \neg Above(x,y,result(Put(b,l),s))]$$

$$\forall b,l,x,y,s[(b\neq l) \wedge (b\neq TABLE) \wedge \forall z \neg On(z,b,s)) \wedge (l=TABLE \vee \forall z \neg On(z,l,s)) \wedge Above(x,y,s) \wedge \neg \exists z[x=b \wedge y=z \wedge z\neq l \wedge \neg Above(l,z,s)] \rightarrow Above(x,y,result(Put(b,l),s))]$$

$$\forall b,l,x,y,s[(b\neq l) \wedge (b\neq TABLE) \wedge \forall z \neg On(z,b,s) \wedge (l=TABLE \vee \forall z \neg On(z,l,s)) \wedge \neg Above(x,y,s) \wedge \neg[(x=b \wedge y=l) \vee \exists z(x=b \wedge y=z \wedge Above(l,z,s))] \rightarrow \neg Above(x,y,result(Put(b,l),s))]$$

Note that introducing a situational argument to a constant symbol whose value can change will produce a function of one argument. For example, the symbol t_o in the Adjust(a', θ',t') schema becomes the function $t_o(s)$ when translating to the situation calculus. Thus, every occurrence of t_o in the α, δ, μ, and π formulas for Adjust(a', θ',t') would be replaced by $t_o(s)$ in the situation calculus axiomatization. This was not done for the symbol *TABLE* in the axioms given above, since *TABLE* does not change interpretation in the blocks world considered here.

It is readily apparent from Figure 1 and the examples above that actions can be described more concisely in ADL than in the situation calculus. Figure 1 also illustrates that the expressiveness of ADL is significantly greater than that of the STRIPS language.

4. Reasoning About the Effects of Actions

Although the situation calculus can be used to reason about the effects of actions described in ADL, it is more efficient to use *regression operators* (Waldinger 1977) for this purpose. Regression operators for actions described in ADL are functions mapping formulas to formulas that produce the necessary and sufficient conditions that must hold prior to the execution of an action in order for a given condition to be true afterward. Stated in the situation calculus, if a^{-1} is the regression operator for action a and φ is a formula, then

$$\forall s[\pi(s) \rightarrow (\varphi(result(a,s)) \leftrightarrow a^{-1}(\varphi)(s))] \qquad (7)$$

where $a^{-1}(\varphi)(s)$ is the formula obtained by adding the situational argument s to all relation, function, and constant symbols in $a^{-1}(\varphi)$ whose values can change, and $\varphi(result(a,s))$ is obtained by introducing $result(a,s)$ as a situational argument in φ.

By composing regression operators, we can determine the necessary and sufficient conditions that must hold in the initial state of a planning problem in order for a given condition to be true at some point in a plan. For example, φ will be true after executing the sequence of actions $a_1 \cdots a_n$ if and only if

$$a_1^{-1} \circ \cdots \circ a_n^{-1}(\varphi)$$

(i.e. $a_1^{-1}(\cdots(a_n^{-1}(\varphi))\cdots)$) is true in the initial state. Because this transformation is possible, all reasoning about the effects of actions on goals can be performed relative to the initial state.

The mathematical basis for constructing regression operators from ADL schemas is provided by their equivalent axiomatizations in the situation calculus. It is easy to show that the state-change and frame axioms for a relation symbol R imply

$$\forall x_1,\ldots,x_n,s[\pi(s)\rightarrow(R(x_1,\ldots,x_n,result(a,s)) \leftrightarrow[\alpha_R(x_1,\ldots,x_n,s)\vee(R(x_1,\ldots,x_n,s) \wedge\neg\delta_R(x_1,\ldots,x_n,s))])] \quad (8)$$

Similarly, the state-change and frame axioms for a function symbol F imply

$$\forall x_1,\ldots,x_n,y,s[\pi(s)\rightarrow(F(x_1,\ldots,x_n,result(a,s))=y \leftrightarrow[\mu_F(x_1,\ldots,x_n,y,s)\vee(F(x_1,\ldots,x_n,s)=y \wedge\neg\exists y\,\mu_F(x_1,\ldots,x_n,y,s))])]. \quad (9)$$

Since first-order logic allows the substitution of equivalent formulas, the theorems above permit any occurrence of an atomic formula that contains the situational argument $result(a,s)$ to be replaced by an equivalent formula involving only the situational argument s. By repeated substitution, we are thus able to transform any formula containing only the situational argument $result(a,s)$ into an *equivalent* formula containing only the situational argument s. The formula containing s therefore defines the necessary and sufficient conditions that must hold prior to executing action a in order for the formula containing $result(a,s)$ to be true afterward. This transformation thus satisfies the definition of a regression operator as expressed in Formula 7.

The following equations define a class of regression operators based on the transformations described above. Other equation sets are also possible; the equations below may therefore be regarded as a minimally sufficient set. In the base case, $a^{-1}(\varphi)$ is given by

$$a^{-1}[TRUE] \equiv TRUE \quad (10a)$$

$$a^{-1}[FALSE] \equiv FALSE \quad (10b)$$

$$a^{-1}[R(x_1,\ldots,x_n)] \equiv \alpha_R^a(x_1,\ldots,x_n) \vee[R(x_1,\ldots,x_n)\wedge\neg\delta_R^a(x_1,\ldots,x_n)] \quad (10c)$$

$$a^{-1}[F(x_1,\ldots,x_n)=y] \equiv \mu_F^a(x_1,\ldots,x_n,y) \vee[F(x_1,\ldots,x_n)=y\wedge\neg\exists y\,\mu_F^a(x_1,\ldots,x_n,y)] \quad (10d)$$

$$a^{-1}[C=y] \equiv \mu_C^a(y)\vee[C=y\wedge\neg\exists y\,\mu_C^a(y)] \quad (10e)$$

$$a^{-1}[x_1=x_2] \equiv (x_1=x_2) \quad (10f)$$

where $x_1,\ldots,x_n$ and y are variables. The symbol '$\equiv$' denotes syntactic equality between formulas. These equations perform the substitutions allowed by the equivalence theorems of first-order logic. Superscripts have been introduced to the α, δ, and μ formulas to emphasize that these formulas will of course be different for different action schemas.

For the above substitutions to be logically valid, the atomic subformulas must contain variable symbols in the same positions as the variables in Axioms 8 and 9. The following equations allow any atomic subformula to be converted to this form:

$$a^{-1}[R(\ldots,\tau,\ldots)] \equiv \exists z\,[a^{-1}[\tau=z]\wedge a^{-1}[R(\ldots,z,\ldots)]] \quad (10g)$$

$$a^{-1}[F(\ldots,\tau,\ldots)=y] \equiv \exists z\,[a^{-1}[\tau=z]\wedge a^{-1}[F(\ldots,z,\ldots)=y]] \quad (10h)$$

$$a^{-1}[\tau_1=\tau_2] \equiv \exists z\,[a^{-1}[\tau_1=z]\wedge a^{-1}[\tau_2=z]], \quad (10i)$$

where

(1) τ denotes the leftmost term that is not a variable in the expressions $R(\ldots,\tau,\ldots)$ and $F(\ldots,\tau,\ldots)=y$.
(2) z does not appear in $R(\ldots,\tau,\ldots)$, $F(\ldots,\tau,\ldots)=y$, or $\tau_1=\tau_2$.
(3) τ_2 is not a variable.

Finally, the following allow Equations 10*a-i* to be applied to all atomic subformulas of a formula:

$$a^{-1}(\neg\varphi) \equiv \neg a^{-1}(\varphi) \quad (10j)$$

$$a^{-1}(\varphi\wedge\psi) \equiv a^{-1}(\varphi)\wedge a^{-1}(\psi) \quad (10k)$$

$$a^{-1}(\varphi\vee\psi) \equiv a^{-1}(\varphi)\vee a^{-1}(\psi) \quad (10l)$$

$$a^{-1}(\varphi\rightarrow\psi) \equiv a^{-1}(\varphi)\rightarrow a^{-1}(\psi) \quad (10m)$$

$$a^{-1}(\varphi\leftrightarrow\psi) \equiv a^{-1}(\varphi)\leftrightarrow a^{-1}(\psi) \quad (10n)$$

$$a^{-1}(\forall x\,\varphi) \equiv \forall x\,a^{-1}(\varphi) \quad (10o)$$

$$a^{-1}(\exists x\,\varphi) \equiv \exists x\,a^{-1}(\varphi) \quad (10p)$$

To illustrate the use of regression operators, suppose that we have constructed the plan Put(B,C) followed by Put(A,B) (i.e., put B on top of C then A on top of B). Suppose that in addition we wish to determine whether this plan achieves Above(A,D). This is accomplished by computing Put(B,C)$^{-1}$[Put(A,B)$^{-1}$[Above(A,D)]] and then comparing the result with what is known about the initial state. From Equation 10 and Formulas 5 and 6,

$$\text{Put}(A,B)^{-1}[\text{Above}(A,D)] \equiv (A=A\wedge D=B)\vee\exists z(A=A\wedge D=z\wedge\text{Above}(B,z)) \vee[\text{Above}(A,D)\wedge\neg\exists z[A=A\wedge D=z\wedge z\neq B \wedge\neg\text{Above}(B,z)]]$$

which simplifies to

$$(D=B)\vee\text{Above}(B,D).$$

Applying Put(B,C)$^{-1}$ to this result yields

$$\begin{aligned}&\mathrm{Put}(B,C)^{-1}[(D{=}B)\vee\mathrm{Above}(B,D)]\\&\equiv\exists z[(FALSE\vee(D{=}z\wedge\neg\exists z FALSE))\\&\qquad\wedge(FALSE\vee(B{=}z\wedge\neg\exists z FALSE))]\\&\quad\vee(B{=}B\wedge D{=}C)\vee\exists z(B{=}B\wedge D{=}z\wedge\mathrm{Above}(C,z))\\&\quad\vee[\mathrm{Above}(B,D)\wedge\neg\exists z(B{=}B\wedge D{=}z\wedge z{\neq}C\\&\qquad\wedge\neg\mathrm{Above}(C,z))]\end{aligned}$$

which simplifies to

$$(D{=}B)\vee(D{=}C)\vee\mathrm{Above}(C,D).$$

Thus, A will be above D after executing the plan if and only if C is above D in the initial state, or D is a synonym for either B or C.

5. Vivid Reasoning

The efficiency with which one can reason about the effects of actions described in ADL depends upon what is known about the initial state. As we have seen, the regression of a formula can be computed quite efficiently through a process of substituting atomic subformulas for other formulas. The cost of comparing a regressed formula to the initial state, on the other hand, can vary. At one extreme, in which our knowledge about the initial state is incomplete and can only be expressed in the form of unconstrained first-order formulas, the comparison will be computationally intractable in the general case. At the other extreme, however, in which we have complete knowledge of the initial state, an efficient vivid style of reasoning can be employed.

If we truly have complete knowledge of the initial state, it should be possible to assign names to all of the relevant objects and to enumerate the relations and functions that hold among them. In that case, the relevant information about the initial state can be encoded as an *algebraic structure*(Shoenfield 1967; Van Dalen 1980). Such a structure has the form

$$M=\langle D;r_1,\ldots,r_n;f_1,\ldots f_m;d_{k_1},\ldots,d_{k_l}\rangle$$

where D is a nonempty set of objects called the *domain* of the structure, $r_1,\ldots,r_n$ are set-theoretic relations on D, $f_1,\ldots f_m$ are set-theoretic functions on D, and $d_{k_1},\ldots,d_{k_l}$ are distinguished elements of D. The truth value of a formula with respect to this algebraic structure (and, hence, the initial state) can then be determined via a standard set of evaluation rules (e.g., Shoenfield 1967; Van Dalen 1980). For example, the algebraic structure

$$BLK=\langle D_{\mathrm{BLK}};\mathcal{ON},\mathcal{ABOVE};\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D},\mathcal{TABLE}\rangle$$

where

$$D_{\mathrm{BLK}}=\{\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D},\mathcal{TABLE}\}$$

$$\mathcal{ON}=\{\langle\mathcal{A},\mathcal{TABLE}\rangle,\langle\mathcal{B},\mathcal{TABLE}\rangle,\langle\mathcal{C},\mathcal{D}\rangle,\langle\mathcal{D},\mathcal{TABLE}\rangle\}$$

$$\mathcal{ABOVE}=$$

$$\{\langle\mathcal{A},\mathcal{TABLE}\rangle,\langle\mathcal{B},\mathcal{TABLE}\rangle,\langle\mathcal{C},\mathcal{D}\rangle,\langle\mathcal{D},\mathcal{TABLE}\rangle,\langle\mathcal{C},\mathcal{TABLE}\rangle\}$$

models a world consisting of four blocks $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ stacked on a table, where blocks $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{D}$ are resting directly on the table and block $\mathcal{C}$ is stacked on top of block $\mathcal{D}$. The $\mathcal{ON}$ relation specifies which objects are on top of which other objects, while the $\mathcal{ABOVE}$ relation is the transitive closure of $\mathcal{ON}$. If we associate the symbol On with the $\mathcal{ON}$ relation, Above with the $\mathcal{ABOVE}$ relation, and A, B, C, D, and $TABLE$ with the distinguished elements $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{TABLE}$, respectively, then the formula $(D=B)\vee(D=C)\vee\mathrm{Above}(C,D)$ from the example in Section 4 will evaluate true with respect to this structure (and, hence, this initial state).

Determining the truth value of a formula by means of evaluation is exactly the kind of vivid reasoning discussed by Levesque (1986). The evaluation rules found in logic text books, however, are designed to be mathematically convenient, not computationally efficient. To obtain the levels of efficiency necessary in practical applications, evaluation techniques such as those used in relational database systems must be employed.

Relational databases and algebraic structures are conceptually very similar. An algebraic structure can in fact be represented in a relational database by creating a database relation for each relation and function in the structure, and a unary relation to store the domain of the structure. The evaluation of a formula can then be expressed as a database query using the *relational calculus* (e.g., Ullman 1982; Wiederhold 1983). The relational calculus employs first-order formulas to define queries, making the transformation of a formula into a query quite straight forward. Query optimization techniques (e.g., Ullman 1982; Wiederhold 1983) can then be employed to process the query with much greater efficiency than with the conventional evaluation rules of first-order logic.

6. Summary and Conclusions

ADL addresses an important issue in knowledge representation and automatic planning, which is to balance the expressiveness of a representational language against its computational costs. The expressive power of ADL approaches that of the situation calculus, yet its computational costs compare favorably to those of the STRIPS language. In particular, when the initial state of a planning problem is completely known, an efficient vivid style of reasoning can be employed that makes use of query optimization techniques developed for relational database systems. Database techniques are not appropriate, however, when the initial state is only partially known. In such instances, a more general form of theorem proving must be used. This increases computational costs; however, the increase is inevitable and will necessarily be present in any representational language (e.g., Levesque 1986).

In addition to its computational properties, ADL has definite notational advantages and it incorporates a frame

assumption that eliminates the need to specify frame axioms. The syntax of ADL is based on the STRIPS operator language and provides a very intuitive means for specifying the effects of actions. From a semantic standpoint, though, ADL is quite different from STRIPS and can be related to a restricted form of the situation calculus. This restricted form allows the appropriate frame axioms to be derived directly from the state-change axioms in a manner that reflects the frame assumption embodied in ADL. According to this assumption, only those changes explicitly mentioned in an ADL schema are presumed to take place. Consequently, one need only describe the changes made when an action is performed, not what remains unaltered.

ADL was originally developed in conjunction with a planning technique suitable for actions with context-dependent effects (Pednault 1985, 1986, 1988a, 1988b). This planning technique is actually independent of ADL and requires only that regression operators be provided for each action. ADL complements the technique by allowing the necessary regression operators to be readily constructed from descriptions of the effects of actions. As a separate formalism, ADL has been found to have desirable computational properties and to be useful in modeling a wide range of actions, including some that involve dynamic processes and the passage of time. These features together with an associated planning technique should make ADL a useful adjunct to the representational arsenal available in automatic planning.

Acknowledgements

I would like to thank Gio Wiederhold for introducing me to relational database theory and Bob Moore for urging me to develop my planning ideas within a STRIPS-like framework. Their combined influence lead me to reexamine the STRIPS framework and develop the ADL language. I would also like to thank Marla and Corinne Babcock for their comments on the numerous drafts of the paper and for helping with its preparation.

References

[Brown 1987] F.M. Brown, *editor. The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop*. Morgan Kaufmann, Los Altos, California, 1987.

[Fikes and Nilsson 1971] R.E. Fikes and N.J. Nilsson. "STRIPS: a new approach to the application of theorem proving to problem solving." *Artificial Intelligence*, Vol. 2, pp 189–208, 1971

[Green 1969] C. Green,. "Application of theorem proving to problem solving." *Proc. IJCAI-69*, Washington D.C., pp 219–239, 1969.

[Hanks and McDermott 1986] S. Hanks and D. McDermott. "Default reasoning, nonmonotonic logics, and the frame problem." *Proc. AAAI-86*, Philadelphia, Pennsylvania, pp 328-333, August 1986.

[Hanks and McDermott 1987] S. Hanks and D. McDermott. "Nonmonotonic logics and temporal projection." *Artificial Intelligence*, Vol. 33, No. 3, pp 379-412, November 1987.

[Harel 1979] D. Harel. *First-order dynamic logic*. Lecture Notes in Computer Science, Volume 68, C. Goos and J. Hartmanis, *editors*. Springer-Verlag, New York, New York, 1979.

[Hayes 1973] P. Hayes,. "The frame problem and related problems in artificial intelligence." In *Artificial and human thinking*, A. Elithorn and D. Jones, *editors*, pp 45–59. Jossey-Bass Publishers, 1973.

[Kautz 1982] H.A. Kautz. *A First-Order Dynamic Logic for Planning*. Masters Thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 1982.

[Kautz 1986] H. A. Kautz. "The logic of persistence." *Proc. AAAI-86*, Philadelphia, Pennsylvania, pp 401-405, August 1986.

[Levesque and Brachman 1985] H.J. Levesque and R.J. Brachman. "A fundamental tradeoff in knowledge representation and reasoning." In *Readings in Knowledge Representation*, H.J Levesque and R.J. Brachman, *editors*, pp 42–70. Morgan Kaufmann, Los Altos, California, 1985.

[Levesque 1986] H.J. Levesque. "Making believers out of computers." *Artificial Intelligence*, Vol. 30, No. 1, pp 81–108, October 1986 (Originally presented as the "Computers and Thought Lecture" at IJCAI-85, Los Angeles, California, August 1985).

[Lifschitz 1986] V. Lifschitz. "Pointwise circumscription: preliminary report". *Proc. AAAI-86*, Philadelphia, Pennsylvania, pp 406-410, August 1986.

[Lifschitz 1987a] V. Lifschitz. "On the semantics of STRIPS." In *Reasoning about actions and plans: proceedings of the 1986 workshop*, M.P. Georgeff and A.L. Lansky, *editors*, pp 1–9. Morgan Kaufmann, Los Altos, California, 1987.

[Lifschitz 1987b] V. Lifschitz. "Formal theories of action (preliminary report)." *Proc. IJCAI-87*, Milan, Italy, pp 966-972, August 1987.

[McCarthy and Hayes 1969] J. McCarthy and P. Hayes. "Some philosophical problems from the standpoint of

artificial intelligence." In *Machine intelligence 4*, B. Meltzer and D. Michie, *editors*, pp 463–502. Edinburgh University Press, Edinburgh, Scotland, 1969.

[McCarthy 1986] J. McCarthy. "Applications of circumscription to formalizing commonsense knowledge." *Artificial Intelligence*, Vol. 28, No. 1, pp 89-118, 1986.

[Pednault 1985] E.P.D. Pednault. "Preliminary report on a theory of plan synthesis." Technical Report 358, Artificial Intelligence Center, SRI International, Menlo Park, California, August 1985.

[Pednault 1986] E.P.D. Pednault. *Toward a Mathematical Theory of Plan Synthesis*. Ph.D. thesis, Department of Electrical Engineering, Stanford University, Stanford, California, December 1986.

[Pednault 1987] E.P.D. Pednault. "Formulating multiagent, dynamic-world problems in the classical planning framework." In *Reasoning about actions and plans: proceedings of the 1986 workshop*, M.P. Georgeff and A.L. Lansky, *editors*, pp 47-82. Morgan Kaufmann, Los Altos, California, 1987.

[Pednault 1988a] E.P.D. Pednault. "Extending conventional planning techniques to handle actions with context-dependent effects." *Proc. AAAI-88*, Saint Paul, Minnesota, pp 55–59, August 1988.

[Pednault 1988b] E.P.D. Pednault. "Synthesizing plans that contain actions with context-dependent effects," *Computational Intelligence* (to appear, 1988).

[Shoenfield 1967] J.R. Shoenfield. *Mathematical Logic*. Addison-Wesley, Reading, Massachusetts, 1967.

[Shoham 1986] Y. Shoham. "Chronological ignorance: time, nonmonotonicity, necessity and causal theories." *Proc. AAAI-86*, Philadelphia, Pennsylvania, pp 389-393, August 1986.

[Ullman 1982] J.D. Ullman. *Principles of Database Systems: Second Edition*. Computer Science Press, Rockville, Maryland, 1982.

[Van Dalen 1980] D. Van Dalen. *Logic and Structure*. Springer-Verlag, Berlin, West Germany, 1980.

[Waldinger 1977] R. Waldinger. "Achieving several goals simultaneously." In *Machine Intelligence 8*, E. Elcock and D. Michie, *editors*, pp 94–136. Ellis Horwood, Edinburgh, Scotland, 1977.

[Wiederhold 1983] G. Wiederhold. *Database Design: Second Edition*. McGraw-Hill, New York, New York, 1983.

What the Lottery Paradox Tells Us About Default Reasoning

David Poole
Department of Computer Science
University of British Columbia
Vancouver, B.C., Canada V6T 1W5
poole@cs.ubc.ca

Abstract

In this paper I argue that we do not understand the process of default reasoning. A number of examples are given which serve to distinguish different default reasoning systems. It is shown that if we do not make our assumptions explicit we get into trouble with disjunctive knowledge, and if we make our assumptions explicit, we run foul of the lottery paradox. None of the current popular default reasoning systems work on all of the examples. It is argued that the lottery paradox does arise in default reasoning and can cause problems. It is also shown that some of the intuitively plausible requirements for default reasoning are incompatible. How different systems cope with this is discussed.

1 Introduction

Default reasoning is the ability to jump to a conclusion based on the lack of evidence to the contrary. Deduction in standard logic does not allow such reasoning; if some proposition follows from a set of axioms, it follows from a superset of the axioms. There have been many proposals for incorporating default reasoning in logic [Reiter, 1980, McCarthy, 1986, Moore, 1985, Delgrande, 1987, Poole, 1988]. I assume we use default reasoning to predict what is true.

In this paper we consider the problem of default reasoning, and discuss different choices that could be made in developing a default reasoning system. A set of examples is presented which indicates that current default reasoning systems do not work properly.

This is argued in two parts. In the first part (section 2), it is argued that if we do not commit to implicit assumptions made (for example, acknowledging that we assumed Tweety is not an emu when we concluded Tweety could fly) we get into trouble. In the second part I show how the lottery paradox arises when we do commit to assumptions.

When considering the lottery paradox, the main intuition I rely on is the "one step default" property: if "birds fly" (however we represent it) is a default and all we know about an individual is that it is a bird (in particular, we don't know it doesn't fly), we conclude it flies. This seems like a minimal property the default "birds fly" should have.

2 Commitment

Suppose we have d as a default, with exception e. The first question I want to consider is whether we should conclude $\neg e$ as a side effect of concluding d.

Consider the classic example of birds flying:

Example 1 Suppose we want to use the default "birds fly", with emus as exceptions. Suppose we know Polly is a bird, and know nothing else about Polly. As "Birds fly" is an assumption, it seems reasonable to conclude Polly flies. Should we conclude Polly is not an emu? There have been three different solutions to this suggested by different systems.

2.1 Non-committal

The first of the possible answers is that we should not conclude d at all. We should rather conclude only the disjunct $d \vee e$. The rationale is that we do not know whether the exception e is true, so we do not know whether d is true. If we cannot say whether e is true, we should not allow any side effect to the value of e.

This is exactly the situation with circumscription [McCarthy, 1986] with the exception being "fixed" during the minimisation.

I would argue this non-committalness loses the very reason for default reasoning: we can never conclude a default, but only the disjunct of the possibilities. We have lost the ability to jump to conclusions. Such a system is not doing default reasoning at all; we have just invented a new syntax for disjunctions.

We can never use the "birds fly" default to do what was originally intended, namely to conclude something flies from just knowing it is a bird. We would instead conclude either the bird is an emu or flies. Somehow we changed the meaning of "birds fly, but emu's are exceptions" to mean the logical statement "birds are either emus or fly". With many exceptions we could only conclude Polly flies if we could prove polly is not an emu, is not a roast duck, is not in the shell, etc.

2.2 Non-commitment

An alternate view is we should conclude default d, but make no commitment as to whether e is true or not. That is we conclude d, and not conclude $\neg e$. This occurs, in autoepistemic logic [Moore, 1985] when we use the formula[1]:

$$\neg L \neg d \wedge \neg L e \Rightarrow d$$

(where the operator L means "know"), to mean that if we don't know d is false and we don't know e is true, conclude d.

Similarly, we can use Reiter's semi-normal defaults (as advocated in [Reiter and Criscuolo, 1981]):

$$\frac{: M(d \wedge \neg e)}{d}$$

to mean if $d \wedge \neg e$ is consistent, conclude d.

These get funny (and I would argue, incorrect) results, because they are being non-committal about the assumptions they are making. They do not allow us to conclude anything about the exception e. Consider the following example:

Example 2 Suppose by default people's left arms are usable, but a person with a broken left arm is an exception, and similarly people's right arms are, by default, usable, but broken right arms are an exception. In Reiter's notation (ignoring variables, which are irrelevant to this example) this is

$$\frac{: M(\textit{left-hand-usable} \wedge \neg \textit{left-hand-broken})}{\textit{left-hand-usable}}$$

$$\frac{: M(\textit{right-hand-usable} \wedge \neg \textit{right-hand-broken})}{\textit{right-hand-usable}}$$

If we know nothing about Matt's left arm, we conclude (correctly as to what we assumed a default was) his left arm is usable. If we know his left arm is broken, we (correctly again) do not conclude his left arm is usable.

Suppose we remember seeing him with a broken left arm or a broken right arm (we can't remember which). We add

$$\textit{left-hand-broken} \vee \textit{right-hand-broken}$$

In this case we cannot conclude he has a broken left arm and so conclude his left arm is usable. We also cannot conclude he has a broken right arm so we conclude his right arm is usable. We thus conclude both his left arm and his right arm are usable.

I would argue that this is definitely a bug, being able to conclude both arms are usable given we know one of his arms is broken. The problem is we have implicitly made an assumption, but have been prevented from considering what other assumptions we made as a side effect of this assumption. Somehow we needed to commit to the implicit assumption that his left arm was not broken when we used the first default.

This problem of disjunctive exceptions is endemic to the use of non-normal defaults[2].

2.3 Commitment to Assumptions

A third alternative is to conclude d, and as a side effect conclude $\neg e$. This reflects the idea that in concluding d, we are assuming e is not true, because if e were true we could not conclude d.

This is what happens in circumscription when we, for the "birds fly" example, minimise "ab", with "emu" varying and specify

$$\forall x(\textit{bird}(x) \wedge \neg \textit{ab}(x) \Rightarrow \textit{flies}(x))$$
$$\forall x(\textit{emu}(x) \Rightarrow \textit{ab}(x))$$

or in Theorist [Poole, 1988] make "$\textit{birds fly}(X)$" a possible hypothesis and specify as facts

$$\forall x(\textit{bird}(x) \wedge \textit{birds fly}(x) \Rightarrow \textit{flies}(x))$$
$$\forall x(\textit{emu}(x) \Rightarrow \neg \textit{birds fly}(x))$$

When we specify Tweety is a bird, we conclude Tweety is not an emu[3]. In the next section I consider the question as to whether such side effects can cause problems.

3 The Lottery Paradox

There is a famous problem which arises if we assume a proposition is false when its probability falls below some threshold. The problem arises because the conjunction of a number of likely propositions make become very unlikely or even impossible. This is known as the lottery paradox [Kyburg, 1961].

Suppose we have a threshold of ϵ. If there is a lottery with $> 1/\epsilon$ tickets, we assume each of these will not win. The conjunction of the assumptions is inconsistent. This is usually translated in probability theory as indicating that commitment is a bad idea.

In this section I show how the lottery paradox naturally arises in default reasoning systems[4] and can

[1] This analysis does not change if we use the more modular abnormality notation or use Gelfond's [1988] methodology for using autoepistemic logic.

[2] [Poole, 1988] shows how the use of preconditions in Reiter's defaults can lead to errors with disjunction. This example shows non normal defaults lead to errors with disjunctive knowledge. It is interesting to note that the simpler, and differently motivated Theorist system corresponds exactly to Reiter's normal defaults without preconditions [Poole, 1988, theorem 4.1].

[3] In both of these systems we conclude $\neg \textit{emu}(c)$ for any constant c in our language that we do not know is an emu.

[4] [Kyburg, 1988, Perlis, 1987] also discuss how the lottery paradox can arise in a default reasoning system, but from a very different perspective.

potentially cause severe problems for current default reasoning systems. I then examine some possible responses to this problem.

Consider the following example where the circumscription convention of using named abnormality is used, as above. Assume all we are told about Tweety is that Tweety is a bird.

We start off by writing the sort of birds we may encounter in our domain and have a formula like[5]:

$$\forall x\ bird(x) \equiv emu(x) \vee penguin(x) \vee hummingbird(x) \vee sandpiper(x) \vee albatross(x) \vee \ldots \vee canary(x)$$

Now add defaults about birds. For each sort of bird that is exceptional in some way we will be able to conclude Tweety is not a bird of that sort.

- We conclude that Tweety is not an emu or a penguin because they are exceptional in not flying.
- We conclude Tweety is not a hummingbird as hummingbirds are exceptional in their size (consider for example the case of making a bird cage for Tweety; we have to make an assumption about the size of birds),
- we conclude Tweety is not a sandpiper as sandpipers are exceptional in nesting on the ground (for example, when bush walking and someone says "look at that bird nest", we have to look somewhere first; we look up by default if all we know is the nest belongs to a bird);
- we conclude Tweety is not an albatross as albatrosses are exceptional in some other way.

If every sort of bird is exceptional in some way, except for, say, the canary, we conclude Tweety is a canary (as we have ruled out all the other alternatives). This may or may not be a bad side effect. When we add the fact canaries are abnormal in being a bright colour, suddenly nothing works. We can no longer conclude Tweety flies! $flies(Tweety)$ is no longer in all minimal models. There is one model in which Tweety does not fly and in which all of the other abnormalities are false.

The problem is that local, seemingly irrelevant information (namely information about how different sorts of birds are abnormal in different ways) can interact to make nothing work. When we follow the advertised way to use these default reasoning systems, we find we get very strange behaviour. For seemingly unrelated statements to interact to produce such disastrous side effects is a bad technical problem.

Unlike McDermott [Hanks and McDermott, 1986, McDermott, 1987], I do not suggest this is evidence to give up on the programme of formalising commonsense reasoning using logic, but rather use this problem to shed more light on the phenomenon we are trying to formalise.

[5]This sort of statement naturally arises in systems where we assume complete knowledge.

4 Possible Responses

There a number of possible responses to this problem:

4.1 Denial

The first response is denial that this problem will ever arise in practice. Unfortunately this is an empirical question and not a theoretical question. We can argue about this forever, but until we actually go and build real systems and find out what does happen, the argument will be as irrelevant as trying to determine how many angels can fit on the head of a pin.

The problem outlined here was discovered by using our Theorist system [Poole et. al., 1987, Poole, 1988], and noticing funny side effects and obscure reasons why we should not predict (membership in all extensions) certain expected outcomes. We are currently building larger systems to determine whether such problems do arise. Unfortunately we will never be able to say this problem does not arise in practice, but only be able to determine it does.

I do not believe the scenario above, considering each type of bird as being exceptional in some way, is so far fetched. I would not be surprised, in a large database, if each subclass of bird is indeed exceptional in some way. All we need for the above problem to arise is some way to determine there is no completely typical individual. Once we can determine this, none of the formalisms (that commit to an assumption) work correctly. In large knowledge bases, not only would I expect such situations to arise, but they would be normal. For example, the "normal" person (who is 175cm tall, has an IQ of 100, has a grade 12 education and has 2.2 children), does not exist, although we may want to make these assumptions so we can point out to others how someone is different to that "normal" person.

Example 3 As a natural example, take the well known default in the legal system namely "people are presumed innocent unless proven guilty", and the knowledge that someone is guilty (as there was a crime committed). This could be represented as

$$\forall x\ \neg ab(innocence, x) \Rightarrow \neg guilty(x)$$
$$\exists x\ guilty(x)$$

For any particular individual we do not conclude they are not guilty. I would not like to be the one to explain to judge Jones that we do not conclude

$$\neg guilty(judge\_jones)$$

but do conclude

$$\neg guilty(judge\_jones) \vee \neg guilty(jack\_the\_ripper)$$

4.2 Technical Patches

I described this as a "technical problem", and as such it seems as though it should have a technical solution. I believe the problem is endemic to current ideas about how defaults work (see section 5 below). There is good evidence to suggest that any solution to the lottery paradox above will not work for the broken arm problem (the structure of both of them is remarkably similar, but we expect different answers). I see this as a challenge to those who like to find technical solutions, but I feel as though the problem is we do not understand the phenomena we are trying to formalise.

Suggestions such as prioritised circumscription [McCarthy, 1986] will not work. There is a symmetry about this example. The side effect will affect whatever is the lowest priority default.

One interesting thing can be seen in this example. If we predict what is in one extension rather than what is in all extensions (following the definition of extension in [Reiter, 1980] or [Poole, 1988]), we find "Tweety flies" is in one extension. If we add the exceptions (emu's are abnormal with respect to flying) as facts, we can also predict Tweety doesn't fly. Poole [1988] suggests this problem (of having the side effect explicit) could be solved by using "constraints" to prune the scenarios without being part of the scenarios. This works if we equate prediction with being in one extension. We can explain Tweety flying, but cannot explain the negation. We can explain each of the other typical properties of birds; we cannot predict the conjunction of the properties. The use of constraints also does not let us conclude both of Matt's hands are usable.

This seems to be a good "technical patch" of the type we were looking for. However, equating prediction with membership in an extension leads to the peculiar property of predicting a proposition and also predicting its negation. Careful structuring of the knowledge base may help this (see section 4.3), but this is not a general solution.

If, instead we equate prediction with membership in all extensions, the use of constraints again does not work. The conjunction of all of the normal assumptions about birds is inconsistent; removing the assumption Tweety is normal with respect to flying is a way to make an extension from which we cannot conclude Tweety flies.

4.3 Breaking Conventions

Let us now consider one sort of knowledge it is claimed defaults capture. This is the idea that default reasoning models a notion of conventional reasoning. The reason "birds fly" is a default is that if I tell you Tweety is a bird, and I do not tell you Tweety cannot fly, I am telling you Tweety can fly. This is the motivation for autoepistemic reasoning [Moore, 1985, section 2].

If we take this meaning of default reasoning seriously, not only does the lottery paradox above not arise, but the multiple extension problem in general does not arise.

According to the defaults as conventions view, the default "birds fly" means if I add knowledge about a particular bird, I must assert it doesn't fly if it doesn't fly. With this convention, if I assert Tweety is a bird, and I do not assert Tweety does not fly, I am saying Tweety flies. If Tweety does not fly I have broken the convention. The knowledge based should be fixed up just as if I had asserted something that is false.

If there are multiple extensions they are mutually inconsistent, so at most one can be true of the world under consideration (the intended interpretation). Thus one of the extensions must be false in the intended interpretation. So either something I added explicitly is false in the intended interpretation, or else there is a default which is not applicable in the world under consideration. In the latter case, to follow our convention, I must tell the system about that exception. Multiple extensions indicate I did not follow the convention.

In the lottery paradox example above, Tweety is exceptional in at least one of the properties, so to follow the convention, I should tell you that property. Thus the lottery paradox example cannot arise. Moreover, none of the multiple extension problems can arise. Multiple extensions are thus not a problem to be solved, but indicate a convention has been broken; we need to patch up our buggy knowledge base rather than solve the multiple extension problem.

Automatically enforcing such constraints is not as difficult as it may, at first, seem. In the Theorist system [Poole et. al., 1987, Poole, 1988], we can maintain a knowledge base with only one extension [Poole, 1989a] by ensuring that:

1. When a new default is added to the knowledge base, if we can explain an instance of the negation of the default and cannot prove that instance, this default introduces multiple extensions. If not, we still only have one extension.
2. When a new fact is added, if we can explain the negation of the fact with a explanation containing more than one default and cannot explain it with a subset of that explanation containing only one default, the new fact introduced multiple extensions.

When we detect we have multiple extensions we can ask the user to debug the knowledge base by cancelling one of the defaults [Poole, 1988]. These detection procedures are, in general, undecidable. However it seems appropriate to assign these to low priority background processes, which report when they find an inconsistency or a multiple extension. Just as people do not immediately (if at all) realise they have been mislead (or lied to), these background processes may or may

not return to report a breaking of a convention.

The importance of this section is that "defaults as conventions" is a consistent view of defaults; whether it corresponds to the use of the term default is a different question.

4.4 We don't understand the Phenomena.

The fourth response is we do not understand the phenomena we are trying to formalise. If we mean some sort of "typically", the response in section 4.3 does not seem to be appropriate. If this is the case we must recognise that the lottery paradox can arise in the formal systems defined so far. If we claim the lottery paradox does not arise in the "commonsense" view of a default, then the formal systems do not capture our normal sense of "default". Thus we do not understand the phenomena we are trying to characterise.

In the next section I show that one intuitive reading of "birds fly" is incompatible with many of the formal models of non-monotonic reasoning.

5 One Step Default Property

The property underlying the intuition in the lottery paradox example is what I call the "one step default property"[6].

I will use the notation "$p(x) \rightarrow q(x)$" is a default to mean "p's are q's by default". No meaning should be placed in this notation. Different systems use different notations and have different semantics. I intend this discussion to include all of these notations.

Definition 1 A default reasoning system has the **one step default property** if whenever "$p(x) \rightarrow q(x)$" is a default and all that is given about constant c is "$p(c)$" (in particular we do not know the truth of $q(c)$), it concludes "$q(c)$".

For example, under this property if I tell you "birds fly", and all I tell you about Tweety is Tweety is a bird, if a system has the one step default property it concludes Tweety flies. This seems like a minimal property "birds fly" should have.

The following theorem puts a constraint on the type of systems with this property.

Theorem 1 A default reasoning system cannot have all of the following properties:

(i) The one step default property.

(ii) If it concludes two answers, it concludes their conjunction. That is, if it concludes "a" and concludes "b", it concludes "$a \wedge b$".

(iii) The ability to represent disjunctive knowledge, and to allow arbitrary (not directly conflicting) defaults.

(iv) It does not conclude anything known to be false[7].

Proof: To prove this it suffices to give one set of inputs which follow the constraints given in (iii). By showing that properties (i) and (ii), lead to a contradiction with (iv), we demonstrate that a system with all four properties cannot exist.

Suppose

$$p(x) \rightarrow q_i(x)$$

is a default for $i = 1..n$, and

$$\forall x \ \neg q_1(x) \vee \neg q_2(x) \vee \ldots \vee \neg q_n(x)$$

is a fact, and we are given

$$p(c)$$

By (i) we conclude each "$q_i(c)$", and by (ii) we conclude their conjunction, which is inconsistent, and so must be false, contravening (iv). □

Given these four intuitive properties are inconsistent, it is interesting to consider which property different systems have given up.

(i) is given up in circumscription [McCarthy, 1986], in any minimal model solution [Shoham, 1987] and systems which require membership in all extensions [McDermott and Doyle, 1980]. This is because they want the expressiveness that property (iii) gives, they need property (ii) by their very nature, and always reject having inconsistent extensions or reducing to no models.

(ii) is given up in many probability-based systems [Neufeld and Poole, 1988, Bacchus, 1989], and in systems which, for prediction, only require membership in one extension [Reiter, 1980, Moore, 1985, Poole, 1988]. These latter systems seem to get the one step default property for the wrong reason, namely by being able to predict a proposition and also predict its negation.

(iii) is given up in inheritance systems [Thomason and Horty, 1988]. These allow (i), (ii) and (iv), however they lack the expressiveness of the richer logic-based formalisms.

(iv) is not given up by any system I know, although it is argued [Israel, 1980, Perlis, 1987, Kyburg, 1988] that commonsense reasoning does indeed require reasoning under inconsistency.

[6] This discussion is in terms of parametrised (open) defaults as is it most natural for this case. However the argument is purely propositional, and covers propositional systems as well as systems allowing defaults with free variables.

[7] We do not want it to be inconsistent if the facts are consistent. This property does not constrain the system at all if the facts given are inconsistent.

The ϵ-semantics of [Pearl, 1988] fits into this analysis in a very interesting way. For this theorem it fails in property (iii). There is no consistent probability assignment for the defaults and facts given in the proof. This could be translated as meaning it solves the problem nicely, but I would claim it means we must treat seriously the semantics saying there are only infinitesimally few exceptions. It shows we cannot use the system if the proportion of exceptions does not have measure zero. In particular this system does not seem appropriate to represent "birds fly", as it is not true there are infinitesimally few birds that don't fly. His semantics means accepting the "convention" view of defaults (section 4.3).

Shoham [1987] rejects the one step default property in his discussion on the lottery paradox. However his discussion indicates that we would not want to write such defaults, but explicitly rejects the view of defaults as autoepistemic statements (section 4.3). Rather than indicating to the user that the knowledge base is inconsistent, he would rather [Shoham, 1987, p. 392] the system decide that the user was not rational in adding the default that each lottery ticket would not win, and so not allow the one step default conclusion.

6 Where to look for a solution

I think there are two areas to look for a solution to this problem: these are in the areas of probability theory, and in comparing logical arguments as to why we should believe some proposition or not.

6.1 Probability

Pearl [1988] and Cheeseman [1985] argue very logically and convincingly that probability theory is the correct way to consider reasoning under uncertainty.

The one step default property is ingrained at the very foundation of probability theory. $p(A|B) = v$ only tells us information about A when all we know is B[8] [Pearl, 1988]. Not unsurprisingly, default reasoning systems based on probability theory (eg. [Geffner, 1988]) end up with different properties than those based on minimal models or other logical formalisms which do not have the one step default property.

According to probability theory the lottery paradox is a problem with commitment to assumptions. The problem is concluding a proposition is true without being certain of the proposition. Instead of concluding Tweety flies we could conclude the probability of Tweety flying is high. The conjunction of the conclusions would have probability zero, but we know we cannot conclude the conjunction of propositions is likely just because the proposition is likely.

One of the promising ideas in this area is to use qualitative probabilities [Aleliunas, 1988], where instead of using numbers we can use more linguistic probability values in a probability algebra. The relationship between this and notions of default reasoning is not clear.

Another promising idea is that of Neufeld [1988], where "birds fly" means the probabilistic statement Tweety being a bird increases our belief in Tweety flying:

$$p(flies|bird) > p(flies)$$

The lottery paradox is overcome by not allowing us to conclude the belief in the conjunction is increased just because belief in each proposition is increased.

6.2 Arguments

The second promising area is to consider the role of logical arguments.

Logic can be seen as the study of arguments; it is the study of when we should believe an argument based on the truth of the premises. A valid logical argument is one in which the conclusion must be true if the premises are true. It has been shown [Poole, 1988] that defaults can be treated as possible hypotheses that can be used in the premise of a logical argument. The defaults are the premises of a logical deduction; we do not defeat the argument, but defeat the premises (by showing they are inconsistent). All of the arguments are standard logical proofs. Multiple extensions indicate there is an argument for a proposition and an argument against a proposition.

A natural way to consider default reasoning is to compare the arguments for and against some proposition. Poole [1989a] shows how membership in all extensions can be seen as a process of dialectics. A goal is in all extensions if and only if there is a set of explanations for the proposition such that there is no scenario inconsistent with all of the explanations. This can be modelled at two agents having an argument; one agent finds arguments for the goal and the other agent tries to find a scenario in which all of the first agent's arguments fall down [Poole, 1989a].

In the example of section 3, given Tweety is a bird, there is a very short argument that Tweety flies (namely because Tweety is a bird and "birds fly"). There is a long convoluted argument saying Tweety does not fly (namely by assuming other normalities of birds which eliminates all other possible types of birds Tweety can be, except for the non-flying ones).

It seems reasonable to view reasoning as a process of evaluating logical arguments (or more precisely the premises of logical arguments), and preferring more direct (in some sense) arguments. There is one consequence of this way to view the lottery paradox. We end up with a direct argument that Tweety flies. We end up with all of the other direct arguments about Tweety. The problem is the conjunction of these assumptions is inconsistent. Although we would predict a number of consequences of our knowledge, we

[8] In particular, if $v \neq 0$, it tells us nothing about the value if $p(A|B \wedge C)$.

may not want to predict the conjunction of these consequences. This is exactly the lottery paradox. We predict any particular lottery ticket is not going to win. When we conjoin many such predictions problems arise.

One of the reasons the lottery paradox example is so persuasive is because of our intuitions about wanting to prefer more specific knowledge [Touretsky, 1986, Thomason and Horty, 1988, Poole, 1985, Geffner, 1988]. One intuition behind specificity is exactly the one step default property (we prefer the one step default that emu's don't fly over the longer argument that emus are birds and birds fly). If this is so, any method that compares extensions or models (without regard to the question being asked) is not going to be the basis for a model of default reasoning incorporating specificity. Either it says "yes" to the conjunction of the predictions (which is inconsistent, and so would predict something known to be false), or it says "no" to flies(Tweety), and so must find very circuitous arguments to defeat the direct implication (which seems antithetical to the notion of preferring more specific knowledge).

This idea of solving specificity problems by comparing logical arguments is pursued further in [Poole, 1989b].

7 Conclusion

Rather than suggesting we give up on logic when we find the default reasoning formalisms do not give the answers I would like, I have argued that we need to reconsider the phenomena we are trying to formalise. The way the lottery paradox can easily arise shows the fragility of current default reasoning systems. I believe the right solution is to consider the role of dialectics; we must compare arguments for and against propositions. However, for logicists to defeat the arguments that probability theory is the appropriate framework in which to view this will not be easy. We both need to understand the problems we are trying to solve.

The other moral of this paper is that we must build systems to see how our reasoning systems work in practice. The instance of the lottery paradox was found while using our Theorist implementation [Poole et. al., 1987]. No one understands what other problems will arise when we start to solve non-trivial problems.

Acknowledgements

Thanks to all of the people at the 2nd International Workshop on Nonmonotonic Reasoning, at Grassau in June 1988 who were involved in sharpening my ideas on this problem, especially John McCarthy, Matt Ginsberg, Michael Gelfond, Brian Haugh, Scott Goodwin, Vladimir Lifschits and Gerd Brewka. Thanks also to Randy Goebel, Eric Neufeld, Paul Van Arragon, Romas Aleliunas, and Greg Provan for valuable discussions on the points raised in this paper.

This research was supported under NSERC grant OGPOO44121.

References

[Aleliunas, 1988] R. Alelianas, "A new normative theory of probabilistic logic", *Proc. CSCSI-88.*

[Bacchus, 1989] F. Bacchus, "A Modest, but Semantically Well Founded Inheritance Reasoner", Tech. Report, University of Waterloo.

[Cheeseman, 1985] Peter Cheeseman. In defense of probability. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 1002–1009, Los Angeles, California, August 1985. International Joint Committee on Artificial Intelligence.

[Delgrande, 1987] J. P. Delgrande, "A first-order conditional logic for prototypical properties", *Artificial Intelligence*, Vol. 33, No. 1, 105-130.

[Gelfond, 1988] M. Gelfond, "Autoepistemic Logic and Formalisation of Commonsense Reasoning: Preliminary Report", in *Proceedings of the 2nd International Workshop on Non-Monotonic Reasoning*, Springer-Verlag, Lecture Notes in Artificial Intelligence, No. 346, pp. 176-186.

[Geffner, 1988] H. Geffner, "On the logic of Defaults", Proc. AAAI-88, Minneapolis.

[Hanks and McDermott, 1986] S. Hanks and D. McDermott, "Default Reasoning, Nonmonotonic Logics and the Frame Problem", *Proc. AAAI-86*, pp. 328-333.

[Israel, 1980] D. J. Israel, "Whats wrong with nonmonotonic logic", *Proc. AAAI-80*, 99-101.

[Kyburg, 1961] Henry E. Kyburg, Jr., *Probability and the Logic of Rational Belief*, Wesleyan University Press, Middletown, 1961.

[Kyburg, 1988] Henry E. Kyburg, Jr., Probabilistic Inference and Non-monotonic Inference, *Proc. Fourth Workshop on Uncertainty in Artificial Intelligence*, pp. 221-228.

[McCarthy, 1986] J. McCarthy, Applications of Circumscription to formalising common-sense knowledge, *Artificial Intelligence*, Vol. 28, pp. 89-116.

[McDermott and Doyle, 1980] D. V. McDermott and J. Doyle, "Non-monotonic logic 1", *Artificial Intelligence*, Vol. 13, pp. 41-72.

[McDermott, 1987] D. McDermott, A Critique of Pure Reason, *Computational Intelligence*, Vol. 3, No. 3, August 1987, pp. 151-160.

[Moore, 1985] R. C. Moore, Semantical Considerations on nonmonotonic logic, *Artificial Intelligence* 25 (1) 75-94.

[Neufeld and Poole, 1988] E. Neufeld and D. Poole, Probabilistic Semantics and Defaults, *Proc. Fourth Workshop on Uncertainty in Artificial Intelligence*, pp. 275-282.

[Pearl, 1988] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, 1988.

[Perlis, 1987] D. Perlis, "On the Consistency of commonsense reasoning", *Computational Intelligence*, Vol. 2, No. 4, 180-190.

[Poole, 1985] D. Poole, "On the Comparison of Theories: Preferring the Most Specific Explanation", Proc. IJCAI85, pp.144-147.

[Poole, 1988] D. Poole, A logical framework for default reasoning, *Artificial Intelligence*, Vol 36, pp. 27-47.

[Poole, 1989a] D. Poole, "Explanation and Prediction: An Architecture for Default and Abductive Reasoning", to appear, *Computational Intelligence.*

[Poole, 1989b] D. Poole, Dialectics and Specificity, in preparation.

[Poole et. al., 1987] D. L. Poole, R. G. Goebel, and R. Aleliunas, "Theorist: a logical reasoning system for defaults and diagnosis", in N. Cercone and G.McCalla (Eds.) *The Knowledge Frontier: Essays in the Representation of Knowledge*, Springer Varlag, New York, 1987, pp. 331-352.

[Reiter, 1980] R.Reiter, "A Logic for Default Reasoning", *Artificial Intelligence*, Vol. 33, pp. 81-132.

[Reiter and Criscuolo, 1981] R. Reiter and G. Criscuolo, "On Interacting Defaults", *Proc. Seventh International Joint Conference on Artificial Intelligence*, pp. 270-276.

[Shoham, 1987] Y. Shoham, "Nonmonotonic Logics: Meaning and Utility", *Proc. IJCAI-87*, pp. 388-393.

[Thomason and Horty, 1988] R. H. Thomason and J. F. Horty, "Logics for Inheritance Theory", in *Proceedings of the 2nd International Workshop on Non-Monotonic Reasoning*, Springer-Verlag, Lecture Notes in Artificial Intelligence, No. 346, pp. 220-237.

[Touretsky, 1986] D. S. Touretsky, *The Mathematics of Inheritance Theory*, Pitman / Morgan Kaufmann.

Three-Valued Formalizations of Non-Monotonic Reasoning And Logic Programming

Teodor C. Przymusinski
University of Texas at El Paso
(teodor%utep.uucp@cs.utexas.edu)

Abstract

We introduce *3-valued extensions* of major non-monotonic formalisms and we prove that the recently proposed *well-founded semantics* of logic programs is equivalent, for *arbitrary* logic programs, to 3-valued forms of McCarthy's circumscription, Reiter's closed world assumption, Moore's autoepistemic logic and Reiter's default theory.

This result not only provides a further justification of the well-founded semantics, as a natural extension of the perfect model semantics from the class of stratified programs to the class of all logic programs, but it also establishes the class of *all logic programs* as a large class of theories, for which *natural forms of all four non-monotonic formalisms coincide*. It also paves the way for using efficient computation methods, developed for logic programming, as inference mechanisms for non-monotonic reasoning.

1 Introduction

The *well-founded semantics* of logic programs has been recently introduced in [Van Gelder *et al.*, 1988] as an extension of the *perfect model semantics* [Apt *et al.*, 1987; Gelder, 1987; Przymusinski, 1987; Przymusinski, 1988b] from the class of *stratified* logic programs to the class of *all* logic programs.

The author showed [Przymusinski, 1989] that the well-founded semantics has properties entirely analogous to the properties of the perfect model semantics. In particular, well-founded models are minimal models of the program, as well as *iterated fixed points* of natural operators, *iterated least models* of the program and *preferred models*, with respect to a natural priority relation. Moreover, the least fixed point definition of the well-founded model leads to a natural notion of *dynamic stratification* $\{S_\alpha\}_{\alpha \leq \delta}$ of an *arbitrary* logic program.

As a result, the well-founded semantics provides an attractive alternative to the traditionally used semantics of logic programs, based on *Clark's completion* of the program [Clark, 1978; Lloyd, 1984; Fitting, 1985; Kunen, 1987], at the same time eliminating serious drawbacks of the latter (see [Przymusinski, 1988b]).

The perfect model semantics, however, has one more very important property. It has been shown (see [Przymusinski, 1988a]) that *for stratified programs* the perfect model semantics is equivalent to suitable forms of all four major formalizations of non-monotonic reasoning in AI – McCarthy's circumscription [McCarthy, 1980; McCarthy, 1986], Reiter's closed world assumption CWA [Reiter, 1978], Moore's autoepistemic logic [Moore, 1985] and Reiter's default theory [Reiter, 1980] – thus establishing a close link between the areas of logic programming and non-monotonic reasoning and describing a relatively large class of theories for which natural forms of different non-monotonic formalisms coincide.

Originally, it appeared that no extension of this result will be possible for classes of logic programs significantly broader than the class of stratified logic programs. The reason appeared to be the fact that all four proposed extensions of the perfect model semantics – the *stable model semantics* [Gelfond and Lifschitz, 1988] (based on autoepistemic logic), the *default semantics* [Bidoit and Froidevaux, 1988] (based on default logic), the *weakly perfect model semantics* [Przymusinska and Przymusinski, 1988] (based on circumscription or or on CWA) and the well-founded semantics (seemingly not based on any specific non-monotonic formalism, but enjoying the best properties of them all and also the only one defined for *all* logic programs) – turned out to lead to different results.

In this paper we show that, in fact, the well-founded model semantics is also *equivalent to suitable forms of all four major formalizations of non-monotonic reasoning*. However, in order to achieve this equivalence, *3-valued extensions of non-monotonic formalisms* are needed, which is natural in view of the fact that the well-founded semantics is, in general, 3-valued. Accordingly, we define such 3-valued extensions of all four non-monotonic formalisms and we prove that the well-founded semantics is equivalent to suitable 3-valued forms of McCarthy's circumscription, Reiter's closed world assumption, Moore's autoepistemic logic and Reiter's default theory.

This result not only provides a further justification of the well-founded semantics, but it also establishes

a large class of theories – namely the class of all logic programs – for which *natural forms of all four non-monotonic formalisms coincide.* It also paves the way for using efficient computation methods, developed for logic programming, as inference mechanisms for non-monotonic reasoning. Since the well-founded semantics has been shown to have a *sound and complete procedural mechanism*, called *SLS-resolution* ([Przymusinski, 1988b; Przymusinski, 1989]; see [Kemp and Topor, 1988; Seki and Itoh, 1988] for a discussion of effective implementations), the equivalence of semantics implies that SLS-resolution can be used as an inference engine for all four non-monotonic formalisms in the class of logic programs. The extended 3-valued formalisations of non-monotonic reasoning are also likely to have other useful applications, extending beyond the class of logic programs.

The paper is organized as follows. In the next section we define 3-valued models of first order theories. In Section 3 we define 3-valued circumscription and we prove the equivalence of circumscriptive and well-founded semantics. In Section 4 we define 3-valued autoepistemic logic and we show the equivalence of autoepistemic and well-founded semantics. In Section 5 we briefly discuss 3-valued semantics based on the closed world assumption and on default theory and their equivalence to the well-founded semantics. The full version of the paper, containing complete proofs, will appear elsewhere.

2 Three-Valued Models

In this section we define 3-valued models of first order theories (cf. [Przymusinski, 1989]). Throughout the paper, we restrict ourselves to Herbrand models, but our definitions can be easily extended to the non-Herbrand case. We first define the language $\mathcal{L}$ of 3-valued first order logic.

The *alphabet* of $\mathcal{L}$ consists of (finite or countably infinite) sets of constant, predicate and function symbols, a countably infinite set of variable symbols, the connectives $\neg, \vee$, and $\rightarrow$, the existential quantifier $\exists$, and the usual punctuation symbols. We assume that the alphabet contains at least one constant and at least one predicate symbol. A propositional symbol is a predicate symbol of arity zero.

The *language* $\mathcal{L}$ consists of all the well-formed first order formulae obtained using the alphabet. The *Herbrand base* $H_{\mathcal{L}}$ of $\mathcal{L}$ is the set of all ground atoms in $\mathcal{L}$. A *sentence* is a formula, which does not contain unbounded variables. A formula is *ground* if it does not contain any variables. A *theory* P is a set of sentences of $\mathcal{L}$.

By a *3-valued interpretation* I of the language $\mathcal{L}$ we mean a pair $< T; F >$, where T and F are disjoint subsets of the Herbrand base $H_{\mathcal{L}}$ of $\mathcal{L}$. The set T contains all ground atoms true in I, the set F contains all ground atoms false in I and the truth value of the remaining atoms in $U = H_{\mathcal{L}} - (T \cup F)$ is undefined. If A is a ground atom from $H_{\mathcal{L}}$ then we write $val_I(A) = \mathbf{t}$ (resp. $val_I(A) = \mathbf{f}$; resp. $val_I(A) = \mathbf{u}$) if A is true (resp. false; resp. undefined) in I. We call $val_I(A)$ the *truth value* of A in I. A 3-valued interpretation of $\mathcal{L}$ is a 2-valued interpretation of $\mathcal{L}$ if all ground atoms are either true or false in I, i.e. if $H_{\mathcal{L}} = T \cup F$.

If $I_s =< T_s, F_s >$, for $s \in S$, are interpretations, then by their *intersection* we mean the interpretation $I =< \bigcap_{s \in S} T_s, \bigcap_{s \in S} F_s >$. Clearly, A is true (false) in I iff it is true (false) in all interpretations I_s.

Using the truth values of ground atoms we can recursively extend the truth valuation val_I to the set of all sentences as follows.

For any sentences S and V we define:

$$val_I(\neg S) = \neg val_I(S),$$

where $\neg \mathbf{t} = \mathbf{f}$, $\neg \mathbf{f} = \mathbf{t}$ and $\neg \mathbf{u} = \mathbf{u}$,

$$val_I(S \vee V) = max\{val_I(S), val_I(V)\},$$

$$val_I(S \rightarrow V) = \begin{cases} \mathbf{t}, & \text{if } val_I(V) \geq val_I(S) \\ \mathbf{f}, & \text{otherwise.} \end{cases}$$

For any formula $S(x)$ with one unbounded variable x define:

$$val_I(\exists x\ S(x)) = max\{val_I(S(A)) : A \in H_{\mathcal{L}}\},$$

where the ordering of truth values is given by $\mathbf{t} > \mathbf{u} > \mathbf{f}$ and the maximum of an empty set of values is defined as **f**.

Definition 2.1 *A 3-valued interpretation I of $\mathcal{L}$ is a* 3-valued model *of a theory P if $val_I(S) = \mathbf{t}$, for all sentences S in P. If M is 2-valued then it is called a 2-valued model of P.*

We can now define the remaining connectives $(\wedge, \Rightarrow, \leftrightarrow, \Leftrightarrow)$ and the universal quantifier $\forall$ in the usual way:

$$S \wedge V \equiv \neg(\neg S \vee \neg V);$$

$$S \Rightarrow V \equiv V \vee \neg S;$$

$$S \leftrightarrow V \equiv (S \rightarrow V) \wedge (V \rightarrow S);$$

$$S \Leftrightarrow V \equiv (S \Rightarrow V) \wedge (V \Rightarrow S);$$

$$\forall x\ S(x) \equiv \neg \exists x\ \neg S(x).$$

Notice that:

$$val_I(S \wedge V) = min\{val_I(S), val_I(V)\},$$

$$val_I(\forall x\ S(x)) = min\{val_I(S(A)) : A \in H_{\mathcal{L}}\},\ \text{etc.}$$

Remark 2.1 *Although the two implication connectives $\rightarrow$ and $\Rightarrow$ coincide in 2-valued logic, they are in general different in 3-valued logic. For example, $val_I(S \rightarrow S) = \mathbf{t}$, regardless of the truth value of S, but $val_I(S \Rightarrow S) = \mathbf{u}$, if $val_I(S) = \mathbf{u}$. This is a reflection of the fact, that in 3-valued logic we have several*

different notions of implication, all of which are natural and applicable in different contexts. Similar remarks apply to the two equivalence connectives $\leftrightarrow$ and $\Leftrightarrow$. For example, in [Fitting, 1985; Kunen, 1987] the equivalence connective "$\leftrightarrow$" represents the equivalence relation needed to build Clark's completion of the program P, while the equivalence connective "$\Leftrightarrow$" is used in P itself.

In the sequel, by a model we will mean a 3-valued model. We will point out those instances when interpretations are 2-valued.

By a *logic program* we mean a theory consisting of universally quantified clauses of the form

$$\forall\,(A \leftarrow B_1 \wedge \ldots \wedge B_m \wedge \neg C_1 \wedge \ldots \wedge \neg C_n)$$

where m, n$\geq$0 and A, B_i's and C_j's are atoms[1] (see [Lloyd, 1984]). Following a standard convention, such clauses will be simply denoted by

$$A \leftarrow B_1, \ldots, B_m, \neg C_1, \ldots, \neg C_n.$$

Observe, that an interpretation M is a model of a program P if and only if for every ground instance

$$A \leftarrow K_1, \ldots, K_m$$

of a program clause we have

$$val_M(A) \geq val_M(K_1 \wedge \ldots \wedge K_m).$$

Clearly

$$val_M(K_1 \wedge \ldots \wedge K_m) = min\{val_M(K_i) : i \leq m\}.$$

Remark 2.2 *Notice, that in the definition of a program clause we use the implication symbol $\leftarrow$ rather than $\Leftarrow$, because the satisfaction of the condition that $val_I(S \leftarrow S) = \mathrm{t}$, regardless of the truth value of S (see Remark 2.1), is essential for logic programming.*

In [Przymusinski, 1989] we showed that the well-founded model of a logic program can be defined as an iterated least point of a natural operetor and we used this definition to introduce a *dynamic stratification* $\{S_\alpha : 0 \leq \alpha \leq \delta\}$ of an *arbitrary* logic program P. The dynamic stratification $\{S_\alpha : 0 \leq \alpha \leq \delta\}$ of P is a decomposition of the set of all ground atoms in $H_{\mathcal{L}}$ into disjoint strata S_α.

3 Three-Valued Circumscription

In this section we define *3-valued (parallel and prioritized) circumscription* and we show that the well founded semantics of logic programs is equivalent to the semantics of 3-valued prioritized circumscription (with respect to priorities determined by dynamic stratification [Przymusinski, 1989]). This result is an extension of earlier results [Lifschitz, 1987; Przymusinski, 1987; Przymusinski, 1988b] showing that the perfect model semantics of *stratified* logic programs is equivalent to the semantics of (2-valued) prioritized circumscription (with respect to priorities determined by standard stratification).

We use a model-theoretic definition of circumscription and we limit our attention to Herbrand models, but our definitions can be easily extended to non-Herbrand models. The only difference between the (model theoretic) definition of 2-valued circumscription – defined in [McCarthy, 1980; McCarthy, 1986; Lifschitz, 1985; Lifschitz, 1986] – and our definition of 3-valued circumscription consists in the fact that the former uses only 2-valued minimal models while the latter uses all 3-valued models of the circumscribed theory.

Suppose that P is a theory over the language $\mathcal{L}$ and suppose that R and Z are two disjoint subsets of the Herbrand base $H_{\mathcal{L}}$ of $\mathcal{L}$. Atoms in R are called *minimized atoms* and atoms in Z are called *variable atoms*. Atoms which are neither in R nor in Z are called *parameters*. We now define 3-valued (R,Z)-minimal models of P.

Definition 3.1 *We will say that a model $M =< T; F >$ is less than a model $M' =< T'; F' >$* modulo (R,Z) *if both models coincide on parameters, if $T \cap R \subseteq T' \cap R$ and $F \cap R \supseteq F' \cap R$ and if at least one of these inclusions is strict. In other words, $M < M'$* mod (R,Z) *if both models coincide on parameters, differ on the set R of minimized atoms and if M has no more true facts about R than M' and M has no less false facts about R than N'. A model M is an* (R,Z)-minimal model *of P if there is no model M' less than M modulo (R,Z).*

Thus (R,Z)-minimal models minimise the set of true atoms in R, while maximizing the set of false atoms in R, and, at the same time, varying atoms in Z and keeping the value of parameters fixed. Naturally, in case of 2-valued models, minimization of the set of true atoms automatically implies maximization of the set of false atoms, thus in this case the above definition coincides with the standard definition.

Any consistent 3-valued interpretation can be viewed as a function from the Herbrand base H_P to the three-element set {f,u,t}, ordered by $f < u < t$. As observed by Van Gelder, a model M is less than or equal to a model M' *modulo (R,Z)* if and only if the functions M and M' coincide on the set of parameters and if the function M is strictly less than the function M' under the usual (pointwise) ordering of functions.

We now give a model-theoretic definition of 3-valued parallel circumscription.

Definition 3.2 *A structure M is called a model of* 3-valued parallel circumscription CIRC3(P;R;Z) of P, *with atoms in R minimized and atoms in Z varied,*

[1] The symbol $\leftarrow$ denotes reversed implication $\rightarrow$.

if and only if M is a 3-valued (R,Z)-minimal model of P.

We now turn to 3-valued prioritised circumscription. Suppose that $\{S_\alpha\}_{0\leq\alpha<\delta}$ are disjoint subsets of the Herbrand base $H_\mathcal{L}$ of $\mathcal{L}$ and suppose that Z is a subset of $H_\mathcal{L}$ disjoint from all the sets S_α. The collection $\{S_\alpha\}$ can be thought of as assigning different *priorities* for minimization to the elements of the Herbrand base, with the highest priority given to the atoms in S_0, the next highest to the atoms in S_1, etc. Elements of Z will be called, as before, variable atoms.

Definition 3.3 *A structure M is called a model of* 3-valued prioritised circumscription $CIRC3(P; S_0 > S_1 > \ldots; Z)$ of P, *with respect to priorities* $S_0 > S_1 > \ldots$ *and with variables Z if and only if for every* $\alpha < \delta$, *M is an* $(S_\alpha, \bigcup_{\beta>\alpha} S_\beta \cup Z)$*-minimal model of P.*

If $Z = \emptyset$ *then* $CIRC3(P; S_0 > S_1 > \ldots; Z)$ *will be simply denoted by* $CIRC3(P; S_0 > S_1 \ldots)$.

It is easy to see that parallel circumscription is a special case of prioritised circumscription. Now we can state the main theorem of this section:

Theorem 3.1 (Equivalence of well-founded and circumscriptive semantics) *Suppose that P is a logic program and* $\{S_\alpha\}_{\alpha\leq\delta}$ *is its dynamic stratification (see [Przymusinski, 1989]). Then, the well-founded model* M_P *of P coincides with the intersection of all models of prioritized circumscription* $CIRC3(P; S_0 > S_1 > \ldots > S_\delta)$.

Observe that – as it was the case for stratified programs – the prioritisation of the Herbrand universe of the program P, and thus the circumscription policy used with the theory P, is automatically determined by the syntactic form of the logic program.

The following example illustrates the differences between 2-valued and 3-valued circumscription.

Example 3.1 *Suppose that a logic program P is given by:*

$$b \leftarrow \neg a \qquad (1)$$
$$c \leftarrow \neg b, p \qquad (2)$$
$$p \leftarrow \neg p. \qquad (3)$$

The dynamic stratification of P (see [Przymusinski, 1989]) is given by:

$$S_0 = \{a\},\quad S_1 = \{b\},\quad S_2 = \{c\},\quad S_3 = \{p\}$$

and the well-founded model M_P *is equal to* $M_0 =< \{b\}; \{a, c\} >$, *i.e. atoms 'a' and 'c' are false, 'b' is true and 'p' is undefined.* M_P *is the intersection of two models of 3-valued prioritized circumscription* $CIRC3(P; S_0 > S_1 > S_2 > S_3)$ *of P, namely* M_0 *and* $M_1 =< \{b, p\}; \{a, c\} >$. *But* M_1 *is the only model of 2-valued prioritized circumscription* $CIRC(P; S_0 > S_1 > S_2 > S_3)$ *of P. Consequently, 'p' is true under 2-valued circumscription.*

4 Three-Valued Autoepistemic Logic

In this section we define 3-valued autoepistemic logic and we show that the well founded semantics of logic programs is equivalent to the semantics of 3-valued autoepistemic logic. This result extends an earlier result [Gelfond, 1987] showing that the perfect model semantics of a *stratified* logic program P is equivalent to the semantics of (2-valued) autoepistemic logic.

We will denote by **L** the autoepistemic *belief symbol* [Moore, 1985], also called the *belief operator*. Propositional symbols of the form **LS**, i.e. propositions whose names begin with the belief symbol **L**, will be called *belief propositions*. By an *objective formula* of a language $\mathcal{L}$ we mean any formula that does *not* contain any belief propositions.

By an *autoepistemic language* we mean any language $\mathcal{L}$ with the property that its alphabet contains a propositional symbol **LS** for any objective sentence S of the language[2]. By an *autoepistemic theory* we mean a theory over an autoepistemic language.

Before proceeding with the definition of a 3-valued autoepistemic extension of an autoepistemic theory P, we first need to look at the 2-valued definition from a slightly different angle. The following is a definition of a standard (2-valued) autoepistemic extension.

Definition 4.1 *[Moore, 1985] An* autoepistemic extension *of an autoepistemic theory P is any theory E(P) satisfying the condition:*

$$E(P) = Cn(P \cup \{\mathbf{L}S : E(P) \models S\} \cup \{\neg\mathbf{L}S : E(P) \not\models S\}),$$

where the S's range over all objective sentences and Cn(W) denotes the set of all logical consequences of a theory W.

One can easily see, that in order to construct an autoepistemic extension of P one only needs to find an interpretation L of belief propositions **LS**, i.e. an assignement of truth values to belief propositions **LS**, so that for all objective sentences S, **LS** is assigned the value true iff S holds in all models M of P, in which the interpretation of belief propositions **LS** is given by L.

Therefore the definition of a (2-valued) autoepistemic extension can be translated into the following equivalent form.

Denote by $L : S \rightarrow \{t, f\}$ any mapping from the set S of all objective sentences of $\mathcal{L}$ into the two element set {t,f} consisting of truth values *true* and *false*. We can think of L as a *2-valued interpretation of belief propositions* **LS**, assigning to **LS** the value true if

[2]Here by **LS** we simply mean a string denoting a propositional symbol, which begins with an **L** followed by the objective sentence S, e.g. the string "**L** $A \wedge B$".

and only if L(S)=t. For a given interpretation L, let $Mod(L)$ denote the set of all 2-valued models of P, in which the interpretation of belief propositions **LS** is given by L.

For any sentence S let:

$$Val_L(S) = \begin{cases} \mathbf{t}, & \text{if } \forall M \in Mod(L),\ M \models S; \\ \mathbf{f}, & \text{if } \exists M \in Mod(L),\ M \models \neg S. \end{cases}$$

Definition 4.2 *By a 2-valued autoepistemic extension of an autoepistemic theory P we mean a pair (P,L), where L is a 2-valued interpretation of belief propositions satisfying the condition:*

$$L(S) = Val_L(S), \text{ for all objective sentences } S.$$

As shown by the following proposition, the above definition is equivalent to the standard one.

Proposition 4.1

- *If E(P) is an (old) autoepistemic extension of P, then clearly E(P) uniquely determines the interpretation $L : \mathcal{S} \to \{t, f\}$ and the pair (T,L) is a (new) autoepistemic extension of P.*
- *If (P,L) is a (new) autoepistemic extension of P, then*

$$E(P) = Cn(P \cup \{\mathbf{L}S : L(S) = t\} \cup \{\neg\mathbf{L}S : L(S) = f\})$$

is an (old) autoepistemic extension of P. Equivalently, $E(P) = \{S : Val_L(S) = t\}$.

We can now define 3-valued autoepistemic extensions of an autoepistemic theory P.

Denote by $L : \mathcal{S} \to \{t, u, f\}$ any mapping from the set $\mathcal{S}$ of all objective sentences of $\mathcal{L}$ into the three element set {t,u,f} consisting of truth values *true*, *undefined* and *false*. We can think of L as a *3-valued interpretation of belief propositions* **LS**, assigning to **LS** the value true if L(S)=t, etc. For a given interpretation L, let $Mod3(L)$ denote the set of all 3-valued models of P, in which the interpretation of belief propositions **LS** is given by L.

For any sentence S let:

$$Val3_L(S) = \begin{cases} \mathbf{t}, & \text{if } \forall M \in Mod3(L),\ M \models S; \\ \mathbf{f}, & \text{if } \exists M \in Mod3(L),\ M \models \neg S; \\ \mathbf{u}, & \text{otherwise.} \end{cases}$$

Definition 4.3 *By a 3-valued autoepistemic extension of an autoepistemic theory P we mean a pair (P,L), where L is a 3-valued interpretation of belief propositions satisfying the condition:*

$$L(S) = Val3_L(S), \text{ for all objective sentences } S.$$

One can show that 3-valued autoepistemic extensions are strictly *more expressive* than 2-valued extensions. To illustrate this point, consider the following simple example.

Example 4.1 *Suppose that the theories P and P' are defined by:*

$$P: \quad a \Leftarrow \neg a$$

$$P': \quad a \leftarrow \neg a.$$

Then both P and P' have exactly one 2-valued autoepistemic extension, given by L(a)=t. On the other hand, although L is still the only 3-valued autoepistemic extension of P, the unique 3-valued autoepistemic extension L' of P' is given by L'(a) = u. Thus 3-valued extensions differentiate between the two theories, which is crucially important if they are to provide the right semantics for logic programs.

Also, 3-valued extensions often exist when 2-valued extensions do not.

Example 4.2 *Suppose that the theory P is given by:*

$$P: \quad a \leftarrow \neg\mathbf{L}a.$$

Then P does not have any 2-valued autoepistemic extension, but its 3-valued extension is given by L(a)=u, reflecting the fact that our beliefs about a are indeterminate.

From now on, unless stated otherwise, by an autoepistemic extension we will mean a 3-valued autoepistemic extension.

Definition 4.4 *Let (P,L) be an autoepistemic extension of P. Sentences S for which $Val3_L(S) = t$ (resp. $Val3_L(S) = f$; resp. $Val3_L(S) = u$) are said to be* believed (resp. disbelieved; resp. undefined) *in (P,L).*

We say that a sentence S is believed (resp. disbelieved) *in P if S is believed (resp. disbelieved) in all autoepistemic extensions (P,L) of P; otherwise we say that S is* undefined. *This terminology will be used only if P has at least one extension.*

Proposition 4.2 *An objective sentence S is believed (resp. disbelieved, undefined) in (P,L) iff* **LS** *is believed (resp. disbelieved, undefined) in (P,L). Moreover, S is disbelieved (resp. believed) in (P,L) iff* $\neg\mathbf{L}S$ *is believed (resp. disbelieved) in (P,L).*

Consequently, an objective sentence S is believed (resp. disbelieved, undefined) in P iff **LS** *is believed (resp. disbelieved, undefined) in P. Moreover, S is disbelieved (resp. believed) in P iff* $\neg\mathbf{L}S$ *is believed (resp. disbelieved) in P.*

Observe that with 3-valued autoepistemic logic we are able to talk about *uncertain beliefs* at the level of *individual extensions*. In particular, a theory with a single extension does not automatically force us to have all of our beliefs fully determined.

By an *autoepistemic logic program* we mean an autoepistemic theory consisting of universally quantified clauses of the form

$$\forall\ (A \leftarrow B_1 \wedge \ldots \wedge B_m \wedge \neg\mathbf{L}C_1 \wedge \ldots \wedge \neg\mathbf{L}C_n)$$

where m, n≥0 and A, B_i's are atoms and C_j's are objective sentences. Observe that autoepistemic logic programs are, in particular, logic programs and therefore such clauses will also be denoted by

$$A \leftarrow B_1, \ldots, B_m, \neg \mathbf{L}C_1, \ldots, \neg \mathbf{L}C_n.$$

In order to show the equivalence between the well-founded semantics of logic programs and the 3-valued autoepistemic semantics, we need to translate standard logic programs into autoepistemic logic programs. We follow Gelfond's approach.

Definition 4.5 *[Gelfond, 1987] Let P be a logic program. The autoepistemic logic program $\hat{P}$, which we call the* autoepistemic translation of P, *consists of all clauses of the form*

$$A \leftarrow B_1, \ldots, B_m, \neg \mathbf{L}C_1, \ldots, \neg \mathbf{L}C_n,$$

for all possible ground instances

$$A \leftarrow B_1, \ldots, B_m, \neg C_1, \ldots, \neg C_n$$

of clauses from P.

Now we can state the main theorem of this section:

Theorem 4.1 (Equivalence of well-founded and autoepistemic semantics) *Suppose that P is a logic program, M_P is its well-founded Herbrand model and $\hat{P}$ is the autoepistemic translation of P. Then for any ground atom A the following holds:*

- *A is true in M_P iff A is believed in $\hat{P}$;*
- *A is false in M_P iff A is disbelieved in $\hat{P}$;*
- *A is undefined in M_P iff A is undefined in $\hat{P}$.*

This result generalises the result from [Gelfond, 1987] stating that the perfect model semantics coincides with the 2-valued autoepistemic semantics (or – equivalently – with the stable model semantics [Gelfond and Lifschitz, 1988]) for *stratified* logic programs. More generally, it also extends the result from [Van Gelder *et al.*, 1988] stating that *2-valued* well-founded models coincide with stable models. Observe, that – as opposed to circumscription – no explicit 'prioritisation' of ground atoms was necessary to obtain the equivalence of the two semantics.

As a byproduct of the main theorem we obtain the following important result:

Theorem 4.2 *Every autoepistemic logic program has at least one 3-valued autoepistemic extension.*

Analogous result is false for 2-valued autoepistemic extensions (see Example 4.2).

Example 4.3 ([Van Gelder *et al.*, 1988]) *Let the program P be given by:*

$$\begin{aligned} a &\leftarrow \neg b \\ b &\leftarrow \neg a \\ p &\leftarrow \neg p \\ p &\leftarrow \neg a. \end{aligned}$$

Thus its autoepistemic translation $\hat{P}$ is given by:

$$\begin{aligned} a &\leftarrow \neg \mathbf{L}b \\ b &\leftarrow \neg \mathbf{L}a \\ p &\leftarrow \neg \mathbf{L}p \\ p &\leftarrow \neg \mathbf{L}a. \end{aligned}$$

The program $\hat{P}$ has two autoepistemic extensions L_1 and L_2:

$$L_1(a) = f,\ L_1(b) = t,\ L_1(p) = t$$

$$L_2(a) = u,\ L_2(b) = u,\ L_2(p) = u.$$

Accordingly, 'a', 'b' and 'p' are undefined, as they should be, according to the well-founded semantics. However, since only the first extension L_1 is 2-valued, in the 2-valued autoepistemic semantics (i.e. in the stable model semantics [Gelfond and Lifschitz, 1988]) 'b' and 'p' are believed and 'a' is disbelieved. This illustrates one of the discrepancies existing between the 2-valued and 3-valued autoepistemic semantics.

Example 4.4 *Let the program P be:*

$$a \leftarrow \neg a$$

and thus its autoepistemic translation $\hat{P}$ is:

$$a \leftarrow \neg \mathbf{L}a.$$

$\hat{P}$ has a 3-valued autoepistemic extension given by L(a)=u. Accordingly, 'a' is undefined in $\hat{P}$, again agreeing with the well-founded semantics. However, no 2-valued extensions exist, and thus the 2-valued autoepistemic semantics is indeterminate. This illustrates another discrepancy existing between the 2-valued and 3-valued autoepistemic semantics.

5 Three-Valued Default Theory and CWA

Similarly, 3-valued extensions of the closed world assumption and of the default theory can be defined and shown equivalent to the well-founded semantics. However, due to space limitation, details will be omitted here and given in the full paper.

We only point out, that due to the close relationship existing between autoepistemic logic and default theory [Bidoit and Froidevaux, 1988; Konolige, 1987], 3-valued default theory can be introduced in a manner analogous to 3-valued autoepistemic logic. Also, due to the close relationship existing between circumscription and the closed world assumption [Gelfond *et al.*, 1986; Gelfond *et al.*, 1988], 3-valued closed world assumption can be defined analogously to 3-valued circumscription.

References

[Apt *et al.*, 1987] K. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–142, Morgan Kaufmann, Los Altos, CA., 1987.

[Bidoit and Froidevaux, 1988] N. Bidoit and C. Froidevaux. General logical databases and programs: default logic semantics and stratification. *Journal of Information and Computation*, 1988. (in print).

[Clark, 1978] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322, Plenum Press, New York, 1978.

[Fitting, 1985] M. Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 2(4):295–312, 1985.

[Gelder, 1987] 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, Los Altos, CA., 1987.

[Gelfond, 1987] M. Gelfond. On stratified autoepistemic theories. In *Proceedings AAAI-87*, American Association for Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1987.

[Gelfond and Lifschitz, 1988] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth Logic Programming Symposium*, pages 1070–1080, Association for Logic Programming, MIT Press, Cambridge, Mass., 1988.

[Gelfond *et al.*, 1986] M. Gelfond, H. Przymusinska, and T. Przymusinski. The extended closed world assumption and its relationship to parallel circumscription. In *Proceedings of the Symposium on Principles of Database Systems*, pages 133–139, ACM SIGACT-SIGMOD, 1986.

[Gelfond *et al.*, 1988] M. Gelfond, H. Przymusinska, and T. Przymusinski. On the relationship between circumscription and negation as failure. *Journal of Artificial Intelligence*, 1988. In print.

[Kemp and Topor, 1988] D. Kemp and R. Topor. Completeness of a top-down query evaluation procedure for stratified databases. In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth Logic Programming Symposium*, pages 178–194, Association for Logic Programming, MIT Press, Cambridge, Mass., 1988.

[Konolige, 1987] K. Konolige. *On the relation between default theories and autoepistemic logic.* Research report, SRI International, 1987.

[Kunen, 1987] K. Kunen. Negation in logic programming. *Journal of Logic Programming*, 4(4):289–308, 1987.

[Lifschitz, 1985] V. Lifschitz. Computing circumscription. In *Proceedings IJCAI-85*, pages 121–127, American Association for Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1985.

[Lifschitz, 1986] V. Lifschitz. On the satisfiability of circumscription. *Journal of Artificial Intelligence*, 28:17–27, 1986.

[Lifschitz, 1987] V. Lifschitz. On the declarative semantics of logic programs with negation. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 177–192, Morgan Kaufmann, Los Altos, CA., 1987.

[Lloyd, 1984] J.W. Lloyd. *Foundations of Logic Programming.* Springer Verlag, New York, N.Y., first edition, 1984.

[McCarthy, 1980] J. McCarthy. Circumscription – a form of non-monotonic reasoning. *Journal of Artificial Intelligence*, 13:27–39, 1980.

[McCarthy, 1986] J. McCarthy. Applications of circumscription to formalising common sense knowledge. *Journal of Artificial Intelligence*, 28:89–116, 1986.

[Moore, 1985] R.C. Moore. Semantic considerations on non-monotonic logic. *Journal of Artificial Intelligence*, 25:75–94, 1985.

[Przymusinska and Przymusinski, 1988] H. Przymusinska and T. Przymusinski. Weakly perfect model semantics for logic programs. In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth Logic Programming Symposium*, pages 1106–1122, Association for Logic Programming, MIT Press, Cambridge, Mass., 1988.

[Przymusinski, 1987] T. Przymusinski. On the declarative semantics of stratified deductive databases and logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216, Morgan Kaufmann, Los Altos, CA., 1987.

[Przymusinski, 1988a] T. Przymusinski. Non-monotonic reasoning vs. logic programming: a new perspective. In D. Partridge and Y. Wilks, editors, *Formal Foundations of Artificial Intelligence*, Cambridge University Press, London, 1988. In print. (Extended abstract appeared in: T. Przymusinski. On the relationship between non-monotonic reasoning and logic programming. In *Proceedings AAAI-88*, pages 444–448, American Association for Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1988.).

[Przymusinski, 1988b] T. Przymusinski. On the declarative and procedural semantics of logic programs. *Journal of Automated Reasoning*, 4, 1988. In print. (Extended abstract appeared in: T. Przymusinski. Perfect model semantics. In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth Logic Programming Symposium*, pages 1081–1096, Association for Logic Programming, MIT Press, Cambridge, Mass., 1988.).

[Przymusinski, 1989] T. Przymusinski. Every logic program has a natural stratification and an iterated fixed point model. In *Proceedings of the Eighth Symposium on Principles of Database Systems*, ACM SIGACT-SIGMOD, 1989. (In print).

[Reiter, 1978] R. Reiter. On closed-world data bases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 55–76, Plenum Press, New York, 1978.

[Reiter, 1980] R. Reiter. A logic for default theory. *Journal of Artificial Intelligence*, 13:81–132, 1980.

[Seki and Itoh, 1988] H. Seki and H. Itoh. A query evaluation method for stratified programs under the extended cwa. In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth Logic Programming Symposium*, pages 195–211, Association for Logic Programming, MIT Press, Cambridge, Mass., 1988.

[Van Gelder *et al.*, 1988] A. Van Gelder, K. Ross, and Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *Proceedings of the Symposium on Principles of Database Systems*, ACM SIGACT-SIGMOD, 1988.

Skeptical Reasoning and Disjunctive Programs *

Arcot Rajasekar[1] **Jorge Lobo**[1] **Jack Minker**[1,2]
Department of Computer Science[1]
and
Institute for Advanced Computer Studies[2]
University of Maryland
College Park, Maryland 20742

Abstract

One of the problems in non-monotonic reasoning is the existence of multiple extensions for a given theory. Certain classes of theories have been identified where a unique extension may be selected and inferences made using such an extension is considered 'natural'. However,there are several cases where there is no other alternative and several or all extensions of a theory have to be considered when performing inferences. In such cases we want to be skeptical, that is we are unwilling to believe some statement without conclusive evidence but we are willing to be uncertain in our beliefs. In this paper, we model skepticism as indefinite information. We first present some examples which show the need for skeptical beliefs and show how such information can be represented naturally by disjunctive programs. We also present a proof procedure which can be used to infer answers from these programs.

1 Introduction

One of the problems in non-monotonic reasoning is the existence of multiple extensions for a given theory [3]. Searching in all extensions of a theory can be computationally expensive. Certain classes of theories have been identified where a unique extension may be selected and inferences made using such an extension is considered 'natural' [6, 21]. These are theories based on Horn clauses which use the concept of stratification to identify a unique extension and then use SLDNF-resolution as the inference mechanism [1, 28]. Prioritized Circumscription has been shown to imply such stratification techniques. However,there are several cases where there is no other alternative and several or all extensions of a theory have to be considered when performing inferences. In such cases we want to be skeptical, that is we are unwilling to believe some statement without conclusive evidence but willing to be uncertain in our beliefs [4]. In this paper we model skepticism as indefinite information. In Horn-based programs, it is difficult to represent such indefinite information, although for some special cases it may be achieved [15]. Extending Horn programs to be disjunctive programs[1] is an attractive alternative since indefinite information is represented naturally as disjunctive clauses. In addition, recent research on the semantics of disjunctive programs encourages the study of disjunctive theories as a tool for knowledge representation [17, 18, 22]. We first present some examples which show the need for skeptical beliefs and show how such information can be represented naturally by disjunctive programs. We also present a proof procedure which can be used to infer answers from disjunctive programs. The proof procedure takes advantage of the structure of the program based on its dependency graph [1] and performs nearly as well as SLD-resolution [27] when the Horn component of the program is larger than its disjunctive component.

2 Non-monotonic Reasoning

Non-monotonic reasoning deals with performing inferences where conclusions made from a set of facts can be retracted when new facts are added to the original theory. The prototypical example is the inference one makes about birds and their flying capabilities. If a person knows that *tweety* is a bird then he or she can infer that tweety can fly. But if a new fact that tweety is a penguin is also made known to that person, then he or she will retract the original conclusion. This form of reasoning, where one *jumps* to conclusion with available facts, or where one uses certain *default rules* to perform reasoning, or where one makes an in-

*This work was supported by the National Science Foundation under grant number IRI-86-09170 and the Army Research Office under grant number DAAG-29-85-K-0-177.

[1] A Disjunctive program clause is similar to a Horn clause but contains a disjunction of atoms in the head. A formal definition is provided in Section 3.1.

ference which is *consistent* with the available facts is called non-monotonic reasoning. Next, we discuss very briefly some of the non-monotonic formalisms found in the literature.

Default Logic: Default Logic was formalized by Reiter [25] and consists of classical first order logic augmented with *default rules*. A default theory is defined as a two-tuple $< T, D >$, where T is a set of closed $wffs$ and D is a set of default rules of the form

$$\frac{\alpha : M\beta_1, \ldots, M\beta_n}{\gamma}$$

where α, β's and γ are closed wffs. The meaning of the default rule is that if α holds and each of the β's can be consistently assumed then infer γ. Reiter also defines the notion of an extension. Informally, an extension consists of the theory T and additional information derived using the default rules and forms a consistent set of inferences. We provide a formal definition of an extension below:

Definition 1 *([25])*
Suppose that $< T, D >$ is a default theory. Let S be a set of closed wffs and $\Gamma(S)$ be the smallest set satisfying the following property:
(i) $T \subseteq \Gamma(S)$
(ii) $\Gamma(S)$ *is closed under logical consequence*
(iii) if R is a default rule in D (as described above) and $\alpha \in \Gamma(S)$, and $\neg\beta_1, \ldots \neg\beta_n \notin S$, then $\gamma \in \Gamma(S)$.
A set of wffs E is an extension of $< T, D >$ iff $\Gamma(E) = E$, i.e. iff E is a fixpoint of the operator Γ.

We illustrate default logic using the tweety-bird problem.

Example 1 *([25])*
$T = \{bird(tweety) \; ; \; \neg fly(X) \leftarrow penguin(X)\}$
$D = \left\{ \dfrac{bird(X) \; : \; Mfly(X)}{fly(X)} \right\}$
Now $E = \{bird(tweety), fly(tweety)\}$ is the only extension of $< T, D >$.

Consider a new theory $T' = T \cup \{penguin(tweety)\}$. Then, $E' = \{bird(tweety), penguin(tweety)\}$ is the only extension of $< T', D >$.
We can see that default logic behaves non-monotonically.

A default theory can have none, one, or more than one extension [25]. We are interested in the case where a default theory has more than one extension.

Non-monotonic Modal Logic: McDermott and Doyle [13] used modal operators to define the non-monotonic formalism. They augmented first order logic with a modal operator M where, for a given wff p, Mp would mean that *maybe* p is consistent with everything believed. The modal operator defined by McDermott and Doyle is different from the one defined in Reiter's default logic. In Reiter's logic the operator M is allowed only as part of default rules and not as part of the first order theory or inferences. In McDermott and Doyle's logic the operator is used in the sentences of the logic and the conclusions. Moore [19] extended the non-monotonic logic of McDermott and Doyle by replacing the *maybe* operator by an operator L (where Lp means that p is *necessary* to be believed). His logic, autoepistemic logic, is more powerful than McDermott and Doyle's. We briefly describe McDermott and Doyle's non-monotonic formalism.

The language $\mathcal{L}$ of the non-monotonic logic consists of the first order logic language augmented with the modal operator M, and allows sentences such as Mp, $\neg Mp$ where p is a first order formula. The logic uses the monotonic inference rules of modus ponens and generalization along with a new rule [13] of the form:

$$If \;\; \forall_T \neg p, \; then \;\; \vdash_T Mp$$

which means that if the negation of a formula is not derivable from T then it may be consistently inferred by the formula. Derivability from the logic is defined as follows:

Definition 2 *([13])*
Let T be the theory and $S \subseteq \mathcal{L}$. Let the operator NM be defined as

$$NM_T(S) = Th(T \cup As_T(S))$$

where $As_T(S)$ is the set of assumptions from S given by

$$As_T(S) = \{Mq \mid q \in \mathcal{L} and \neg q \notin S\} - Th(A)$$

and where $Th(R)$ is defined as $Th(R) = \{p \mid R \vdash p\}$.
S is a fixpoint when $NM_T(S) = S$
The set of provable formulas, TH(T), is given by the intersection of all fixpoints of NM_T.

Example 2 *([13])*
The tweety-bird problem can be encoded in McDermott and Doyle's logic as:
$T = \{bird(tweety); \; penguin(X) \rightarrow \neg fly(X); \; bird(X) \wedge M(fly(X)) \rightarrow fly(X)\}$
We have that $TH(T) = \{bird(tweety), \; fly(tweety), \; Mfly(tweety)\}$

Consider the new theory $T' = T \cup \{penguin(tweety)\}$. Then, $TH(T') = \{bird(tweety)\}$.

A non-monotonic theory can have none, one, or more than one fixpoint.

Circumscriptive Logic: Circumscriptive theories perform default inferences by axiomatizing the notion of *minimal entailment* from a first order theory. McCarthy introduced predicate circumscription [12] which deal with minimizing the extent (set of values where the predicate is true) of all the predicates in the theory and extended it to prioritized circumscription [11] where the extent of one particular predicate can be minimized without minimizing the extent of other predicates. Minker and Perlis [16] introduced the notion of protected circumscription, where a predicate is restricted to be minimized for some values, and Lifschitz [7] extended prioritized circumscription to pointwise circumscription, where a predicate is minimized one point at a time allowing one to determine if a predicate is true for a possible value. Here, we describe circumscriptive non-monotonic formalism using McCarthy's prioritized circumscription.

Definition 3 *([11])*
Let a tuple P of predicate variables be broken into disjoint parts $P^1, P^2, \ldots, P^k$. Let p^i, q^i be tuples of predicate variables similar to P^i, and let p, q stand for $p^i, \ldots, p^k$ and $q^i, \ldots, q^k$ respectively. Then define,

$$p \preceq q \equiv \bigwedge_{i=1}^{k} (\bigwedge_{j=1}^{i-1} p^j = q^j \Rightarrow p^i \leq q^i)$$

where $\forall P\, P'$, $P \leq P' \equiv \forall x, E(P,x) \Rightarrow E(P',x)$, where $E(P,x)$ is a wff in which P and a tuple of x of variables occur free. Then, the circumscription of a theory A by minimizing the predicates in P^1 at a higher priority than those of P^2 and P^2 at a higher priority than P^3 and so on, is defined as $Circum(A, P^1 > \ldots > P^k; P) = A(P) \wedge \neg\exists p(A(p) \wedge p \prec P)$.

Example 3 *Let $A = \{\forall x,\ P_1(x) \vee P_2(x)\}$ be a first order theory with predicates P_1 and P_2. We want to circumscribe the theory by first minimizing P_1 before P_2. Then,*

$$Circum(A, P^1 > P^2; P^1, P^2) = \forall x, \neg P^1(x) \wedge \forall x, P^2(x)$$

That is, the extent of P_1 is made as small as possible (in this case identically false).

In the case of the tweety-bird problem, we can have two predicates, *fly* and *ab* where the *ab* predicate denotes an abnormality to flying (such as being a penguin). By minimizing *ab* first with respect to fly, we reach the same conclusions as reached by the other two non-monotonic formalisms.

We can characterize the minimal entailment provided by circumscription using minimal models. When we use predicate circumscription to circumscribe a theory T with respect to a predicate P, a formula F is minimally entailed by the circumscribed theory if and only if it is true in all models of T which are minimal in P [12]. There can be more than one model which is minimal in P. When we use prioritized circumscription with priority order, $P^1 > \ldots > P^k$, the formula F is minimally entailed if and only if F is true in all models of T which are minimal according to the priority relationship of the predicates. Przymusinski [21] provides a definition for such models and calls them preferred models. There can be more than one preferred model for a given prioritized circumscription of a theory.

3 Horn Programs

A *general Horn program*[2] consists of a finite set of program clauses of the form

$$A \leftarrow B_1 \wedge \ldots \wedge B_m \wedge \neg B_{m+1} \wedge \ldots \wedge \neg B_n \quad n \geq 0$$

where the As and Bs are atoms. There is a procedural interpretation associated with this set of clauses or rules. Roughly speaking, this interpretation says that to solve a positive literal A we have to solve each of the literals occurring in $B_1 \wedge \ldots \wedge B_m \wedge \neg B_{m+1} \wedge \ldots \wedge \neg B_n$. On the other hand, if we want to solve a negative literal $\neg A$ we attempt to solve A and if we are not able to solve it we can assume $\neg A$ solved, otherwise we assume no solution for $\neg A$. The rule to solve negative literals is know as the *Negation as Failure* rule [2]. This interpretation forms the core of SLDNF-resolution, a proof-procedure to answer queries in general Horn theories. There are some semantic problems when we apply the Negation as Failure rule to any general program and some syntactic restrictions have to be imposed on the program to obtain an amenable semantics. The discussion of these problems is outside the scope of this paper. Nevertheless, we present here the class of programs where SLDNF-resolution is sound. The Herbrand Base of a program P, $HB(P)$, and ground instances are defined as in [8].

Definition 4 ([21]) *A program P is locally stratified if there exists a partition $H_1, \ldots, H_\alpha, \ldots$ of $HB(P)$ where $\alpha < \beta$ for some countable ordinal β, so that for each ground instance of a rule*

$$A \leftarrow B_1 \wedge \ldots \wedge B_m \wedge \neg B_{m+1} \wedge \ldots \wedge \neg B_n$$

in P, if A belongs to a given partition H_k, then

1. *All the positive ground atoms B_i, $1 \leq i \leq m$ belong to $\bigcup\{H_j : j \leq k\}$.*
2. *All the negated ground atoms B_i, $m+1 \leq i \leq n$ belong to $\bigcup\{H_j : j < k\}$.*

Any partition satisfying 1 and 2 is called a local stratification of P. Each element H_α is called a stratum.

The problem with locally stratified programs is that the property of local stratification is not decidable.

[2]Sometimes called normal programs.

Hence, we present a weaker definition of stratification which covers a smaller class of programs which is decidable.

Definition 5 ([1]) *A program P is stratified if there exists a partition $P_1, \ldots, P_t$ of the predicates of P, so that for each rule*

$$A \leftarrow B_1 \wedge \ldots \wedge B_m \wedge \neg B_{m+1} \wedge \ldots \wedge \neg B_n$$

in P, if A belongs to a given partition P_k, then

1. *All the positive atoms B_i, $1 \leq i \leq m$ belong to $\bigcup\{P_j : j \leq k\}$.*
2. *All the negated atoms B_i, $m+1 \leq i \leq n$ belong to $\bigcup\{P_j : j < k\}$.*

Any partition satisfying 1 and 2 is called a stratification of P. Each element in the partition is called a stratum.

There is a strong relationship between general Horn programs and the different models of non-monotonic reasoning. Some of these results are summarized in the next section. A discussion of this can be found in [20].

4 Circumscription and Horn Programs

In Section 2, we saw that default theories can have more than one extension, and that non-monotonic logic theories can reach more than one fixpoint and that circumscription can lead to more than one minimal model. All these cases are variations of the same problem and we call it the *multiple extension problem* [3]. Non-monotonic reasoning systems solve this problem in different ways. In this section, we show how this problem is taken care of by default logic, non-monotonic logic systems and circumscription. We also show, through examples, that these approaches might not be feasible in some cases and propose a solution which is different from those taken by them.

Consistency-based approaches (McDermott and Doyle's Non-monotonic Logic [13] and Reiter's Default Logic [25]) choose a particular extension and use it to perform inferences. Circumscriptive theories (McCarthy's Predicate Circumscription [12]) include all extensions and infer positive and negative information when they are true in all the extensions. But such an approach leads to expensive computation (searching all minimal models) and hence techniques have been identified [11, 7] to confine searching to only one extension for certain classes of theories.

Reiter [24] has shown that for definite Horn programs, circumscription leads to a single extension. For other classes of programs, circumscriptive theories (Prioritized Circumscription [11]) adopt a priority order on predicates, such that a predicate with lower priority is minimized first. Lifschitz [5] has shown that Prioritized Circumscription leads to a single extension when the program is stratified Horn [1]. Przymusinski has also extended prioritization to the domains of the predicates, and his theory, leads to a single extension when the program under consideration is locally stratified [21]. PROLOG, using SLDNF-resolution, is an efficient and easy way to compute Prioritized Circumscription for stratified and locally stratified programs.

The technique of prioritization allows the selection of one out of multiple extensions based on the syntax of a program (as in the case of stratification and local stratification) or on some user-defined priority order. The advantage of this technique is that the computation is simpler than for predicate circumscription and the extension chosen is 'natural' [6, 21].

5 Disjunctive Programs and Multiple Extensions

We are concerned with the types of programs where the selection of an extension through prioritization would be unnatural and contrived. That is, the theory may not be amenable to stratification and a user definition of a priority order might not be possible. Consistency based theories would still work, but with similar problems. The problem of conflicting default rules is ubiquitous in work on non-monotonic reasoning [3].

Example 4 *(Nixon Diamond [26] - Interacting Defaults)*
Consider the problem definition in first-order logic:

$$P(X) \leftarrow Q(X) \wedge \neg H(X).$$

Normally, Quakers are Pacifists.

$$H(X) \leftarrow R(X) \wedge \neg P(X).$$

Normally, Republicans are hawks (non-pacifists).

$$Q(n) \wedge R(n).$$

Nixon is a Quaker and a Republican.

There are two extensions and it is not easy to find which is more powerful than the other. A solution is to include the disjunctive rule $H(X) \vee P(X) \leftarrow R(X) \wedge Q(X)$. which represents one's belief and includes both extensions. We are then able to come to the skeptical inference that Nixon is either a Hawk or a Pacifist, without having to choose among them. This skepticism results since Nixon, represented by constant n, is both a Quaker and a Republican. For an individual known only to be a Quaker, we can conclude that the individual is a Pacifist using the negation as failure rule.

Example 5 *(Database Problem (Chap. 1 [3]) - Non-Stratifiability)*
Suppose we are told that PanAm is starting a new low-fare service from San Francisco to the East Coast, but has not yet decided whether the destination will be New York or Washington. This information can be encoded by the rule $flight(sf, ny, pa000) \vee flight(sf, w, pa000)$. *We cannot encode this either as* $flight(sf, ny, pa000) \leftarrow \neg flight(sf, w, pa000)$ *or as* $flight(sf, w, pa000) \leftarrow \neg flight(sf, ny, pa000)$ *since there is no motivation in choosing either of them. Again, a disjunctive or indefinite representation of the information is needed.*

Example 6 *(Diagnostic Systems - Temporal Skepticism)*
Consider the rule that

$$resistor(X) \wedge ab(X) \rightarrow open(X) \vee short(X).$$

From an observed abnormality, ab(r), for a resistor r, we can infer that r is either open or short. When further evidence shows that r is not short we can conclude that open(r). Choosing an extension randomly or prioritizing the predicates open and short might lead to temporal inconsistency with new evidence. Again an indefinite semantics is needed.

Example 7 *(Abductive Reasoning)*
Abductive reasoning tries to generate explanations about the state of the environment. The following rule is a typical abductive rule: if we know that Q is true and we also know $P \rightarrow Q$ *then we assume P. Consider the following rules in a medical expert system.*

$$F(X) \leftarrow P(X).$$

Normally, fever is caused by Pneumonia.

$$F(X) \leftarrow C(X).$$

Normally, fever is caused by Common Cold.

In an abductive system we should be able to infer that when there is a fever it is caused either by pneumonia or by common cold. After applying the abductive rule one has to be able to reason with information like $P(X) \vee C(X)$. *Again, disjunctive inference is useful and relevant when compared to a less skeptical inference which assumes directly* $P(X)$ *is true or* $C(X)$ *is true but both are not true.*

In three of the examples above, prioritization (stratification) is not possible. In cases like these, we might want a system to follow a "skeptical" reasoning strategy—according to which, at times, it is willing to refrain from drawing definite conclusions. This kind of strategy has been explored in another context in [4]. Another way of looking at this is that we have to protect a predicate X from minimizing predicate Y. That is, if we want to minimize we have to minimize X and Y together (In Example 1, we have to minimize R and H together.) Our proposal is to model skepticism, in the current context, as indefinite information, thereby circumventing the problem of choosing between conflicting defaults. Using indefinite clauses as the way to represent knowledge solves the multiple extension problem by including all extensions in the deduction and such a theory is in compliance with predicate circumscription. In Horn-based programming languages, such as PROLOG, it is difficult to represent such indefinite information. In our current work, we have been studying disjunctive logic programs [17, 18], and believe that these provide a promising vehicle for representing indefinite data. Based on this work, we present in the next section a procedure for answering queries in such non-stratifiable disjunctive theories.

Przymusinski has developed a general algorithm which can be used with first order theories and provides circumscriptive inferences. But his algorithm is computationally expensive. In our approach we consider theories that contain both definite and indefinite clauses. We identify a new way of partitioning the program based on its dependency graph and develop a query answering procedure.

6 SLOP-resolution

The basic problem with reasoning in the presence of indefinite information is that making deductions is much more expensive than with definite or Horn theories. We present a proof procedure for answering queries in disjunctive programs [17]. Although we restrict our attention to disjunctive programs the deduction mechanism is still complex. However, if we assume that a large part of the theory is definite and only a small part is disjunctive we can make use of efficient algorithms for the definite part. The proof procedure, called SLOP-resolution, described below, is a variation of SLD-resolution [27]. SLOP-resolution partitions a program such that if a partition contains only Horn-rules the deduction procedure essentially reduces to SLD-resolution. If the partition contains disjunctive rules SLOP-resolution tries to minimize the transfer of information between the partitions. Section 3.1 contains some basic definitions. In Section 3.2 we describe SLOP-resolution and present an example.

6.1 Preliminary Definitions

We consider a *disjunctive logic program, P,* to consist of a finite set of program clauses of the form

$$A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m \quad n \geq 1,\ m \geq 0$$

where the As and Bs are atoms. The expression on the left hand side of the implication sign is called the

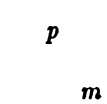
head and the one on the right hand side is called the *body* of the clause. An indefinite or *disjunctive clause* is a program clause with more than one atom in the head, i.e., $n > 1$. An *assertion* or *positive* clause is a program clause which has no body. We use either the notation $C = A_1 \vee \ldots \vee A_p$ or $C = \{A_1, \ldots, A_p\}$ to represent a clause. This allows us to consistently use the relations $\in$ and $\subseteq$ between atoms and clauses. A goal is of the form $\leftarrow C_1, \ldots, C_n$, $n \geq 0$, where the C's are positive clauses. A *definition* of a relation symbol R is the subset of P consisting of all clauses where R occurs in the head. R is disjunctive in P if R occurs in the head of a disjunctive clause. A relation Q **refers** to a relation R if there is a clause in P with Q in the head and R either in the head or the body. Given a positive clause $C = A_1 \vee \ldots \vee A_p$, C θ-subsumes a clause D if θ is the most general unifier for $\{A_1 = D_1, \ldots, A_p = D_p\}$ where $D_1 \vee \ldots \vee D_p$ is a subclause of D.

Definition 6 *The dependency graph of a program P is the directed graph representing the relation* **refers**. *Let $S_1, \ldots, S_n$ be an enumeration of the strongly connected components*[3] *of the dependency graph of P. We say that a partion S_i is disjunctive if there is a disjunctive relation symbol that belongs to S_i otherwise we say it is definite. The partition function $\rho_P : \{A(t_1, \ldots, t_l) | \text{ atoms in } P\} \rightarrow \{S_1, \ldots, S_n\}$ is defined as follows: $\rho_P(A(t_1, \ldots, t_l)) = S_i$ iff $A \in S_i$.* □

Example 8 *The following is a disjunctive program, the dependency graph and its strongly connected components:*

$$P = \{t(X) \leftarrow p(f(X)),\ p(X) \leftarrow m(X),\ p(f(X)) \leftarrow q(X),\ q(X) \leftarrow m(f(f(X))),\ q(X) \leftarrow p(X),\ m(0) \vee m(f(f(X)))\}$$

Dependency Graph:



Strongly-connected components:

$S_1 = \{m\}$, $S_2 = \{p, q\}$, $S_3 = \{t\}$. □

Complete proof procedures for non-Horn theories that use resolution based on model elimination [10] are highly expensive due to ancestry resolution and factoring. Ancestry resolution requires one to keep track of deduction steps during the proof since some intermediate steps might be needed to complete the proof. This kind of deduction is both time consuming since a test is needed to determine if ancestry resolution or factoring is applicable and wasteful of space needed for bookkeeping. In the case of disjunctive programs, ancestry resolution can be characterized using dependency graphs. For this, we extend the relation **refers** to the strongly connected components of P. Given two strongly connected components S, S', we say that S **strongly refers** to S' if there is a relation symbol $R \in S$ and a relation symbol $Q \in S'$ such that R **refers** to Q. This new relation forms a partial order among the strongly connected components. The main result is that we can isolate ancestry resolution inside each strongly connected component and between immediate successors with respect to the refer relation. In the previous example, if we want to prove something about m or q we do not need ancestry information about t. This is the key point used in the proof procedure described in the next section.

6.2 Proof Procedure

In this section we present a refutation proof procedure similar to SLD-resolution to handle disjunctive programs. The soundness and completeness proof can be found in [9].

Definition 7 *Let P be a disjunctive logic program, G be a goal. Assume that $\leq$ is the reflexive and transitive closure of the relation* **strongly refer**. *An* SLOP-derivation *from P with top-goal G consists of a (possibly infinite) sequence of goals $G_0 = G$, $G_1, \ldots$, such that for all $i \geq 0$, G_{i+1} is obtained from $G_i = \leftarrow C_1, \ldots, C_m, \ldots, C_k$ as follows:*

(1) C_m is a clause in G_i. (C_m is called the selected clause)
(2) $C \leftarrow B_1, \ldots, B_q$ is a program clause in P
(3) C θ-subsumes C_m.
(4) G_{i+1} is equal to
$\leftarrow (C_1, \ldots, C_{m-1}, B_1 \vee D_1, \ldots, B_q \vee D_q, C_{m+1}, \ldots, C_k)\theta$ *where*
D_i is a subset of C_m that contains all the atoms A in C_m such that either B_i **refers** *to A or $\rho_P(A) \leq \rho_P(B_i)$.* □

Definition 8 *An* SLOP-refutation *from P with top-goal G is a finite SLOP-derivation of the null clause* □ *from P with top-goal G.* □

In an SLD-derivation step the selected atom is replaced in the goal by the "correct" instantiation of the body of a clause in the program. In our case, we attach a subset of the selected clause in part to transmit the ancestry information that might be needed by the following subgoals. Hence, the reduction in ancestry resolution will be reflected in Step (4) of an SLOP-derivation

[3] Two elements a and b are in the same strongly connected component if and only if there is a path between a and b and a path between b and a.

where some of the atoms in C_m are not inherited by the D_i's. The following is an example of an SLOP-derivation.

Example 9 *Assume the following is part of a dependency graph of program P*

$$m \leftarrow t \leftarrow p \leftarrow q \rightleftharpoons s$$

Suppose now, we are proving the goal $\leftarrow t \vee m \vee s$ *and we apply a derivation step using the rule* $t \leftarrow p$. *The resulting goal is* $\leftarrow p \vee t \vee s$. *Here,* $p = B_1$ *and* $D_1 = t \vee s$. *We keep t since t* refers *to p and s since* $\rho_P(s) \leq \rho_P(p)$. *m is removed since it does not precede p and p does not refer to it.* □

Example 10 *Let P be as in Example 5. An SLOP-refutation for the goal* $\leftarrow t(0)$ *is given below*

(1) $\leftarrow \underline{t(0)}$
using $t(X) \leftarrow p(f(X))$
(2) $\leftarrow \underline{p(f(0))} \vee t(0)$
using $p(f(X)) \leftarrow q(X)$
(3) $\leftarrow \underline{q(0)} \vee p(f(0))$
using $q(X) \leftarrow m(f(f(X)))$
(4) $\leftarrow m(f(f(0))) \vee \underline{q(0)} \vee p(f(0))$
using $q(X) \leftarrow p(X)$
(5) $\leftarrow \underline{p(0)} \vee m(f(f(0))) \vee q(0) \vee p(f(0))$
using $p(X) \leftarrow m(X)$
(6) $\leftarrow \underline{m(0)} \vee p(0) \vee m(f(f(0))) \vee q(0) \vee p(f(0))$
using $m(0) \vee m(f(f(X)))$
(7) □

Notice how from the second to the third step of the derivation $t(0)$ is removed from the goal. This is because q does not refer to t and $\rho_P(t) < \rho_P(q)$.

Theorem 1 ([9]) *For a disjunctive program P and a goal* $\leftarrow C_1, \ldots, C_n$, *there is an SLOP-refutation from P with top-goal G iff* $\forall C_1 \wedge \ldots \wedge C_n$ *is a logical consequence of P.* □

In an SLOP-derivation stronger conditions can be imposed on the D's in Step 4 to decrease the length of the clauses after each derivation without losing the completeness of the proof procedure. Also, θ-subsumption between clauses is not unique. This introduces a new nondeterminism not present in SLD-resolution. There are some restrictions that can be used to constrain the number of choices. A description of these aspects can be found in [9].

6.3 Discussion

SLOP deals with positive information and only positive clauses are deducible. We believe that it is possible to extend SLOP-resolution to handle negative goals. The Generalized Closed World Assumption (GCWA) [14] and the Weak GCWA (WGCWA) [22] provide two alternative approaches to do this. In [22], Lobo, Minker and Rajasekar present a proof procedure to compute negative information using the WGCWA which is similar to SLDNF-resolution [2] and is easily adapted to SLOP-resolution. New results have been obtained on the semantics of general disjunctive programs [23] (disjunctive program with negative literals in the bodies of the clauses.) A means was suggested by the work on stratified Horn programs [1, 28]. Based on the theory of non-monotonic operators developed by Apt, Blair and Walker [1], Rajasekar and Minker describe a sequence of non-monotonic operators to characterize a declarative semantics (fixpoint semantics) for *stratified disjunctive programs* [23]. An alternative characterization of this semantics based on the GCWA is also presented. Proof-procedures for this class of programs are currently under investigation. With these extensions the theory of disjunctive programs would embody both skeptical and prioritized knowledge.

References

[1] 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 Pub., Washington, D.C., 1988.

[2] K. L. Clark. Negation as Failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322, Plenum Press, New York, 1978.

[3] M. L. Ginsberg, editor. *Readings in Nonmonotonic Reasoning.* Morgan Kaufmann Pub., 1987.

[4] J. Horty, R. Thomason, and D. Touretzky. *A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks.* Technical Report CMU-CS-87-175, Computer Secience Department, Carnegie Mellon University, 1987.

[5] V. Lifschitz. Computing circumscription. *Proc. Ninth International Joint Conference on Artificial Intelligence*, Morgan Kaufman Publishers, Inc. 121–127, 1985.

[6] V. Lifschitz. On the declarative semantics of logic programs with negation. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, 1988.

[7] V. Lifschitz. Pointwise circumscription: a preliminary report. *AAAI*, 406–410, 1986.

[8] J.W. Lloyd. *Foundations of Logic Programming.* Springer–Verlag, 1984.

[9] J. Lobo, A. Rajasekar, and J. Minker. Layered Approach for Disjunctive Programs. In preparation.

[10] D.W. Loveland. *Automated Theorem Proving: A Logical Basis*. North-Holland Publishing Co., 1978.

[11] J. McCarthy. Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence*, 28(1):89–116, 1986.

[12] J. McCarthy. Circumscription - a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1 and 2):27–39, 1980.

[13] D. McDermott and J. Doyle. Nonmonotonic Logic I. *Artificial Intelligence*, 13:41–72, 1980.

[14] J. Minker. On indefinite databases and the closed world assumption. In *Lecture Notes in Computer Science 138*, pages 292–308, Springer-Verlag, 1982.

[15] J. Minker and D. Perlis. Computing protected circumscription. *Journal of Logic Programming*, 2(4):235–249, December 1985.

[16] J. Minker and D. Perlis. Protected circumscription. *Proc. Workshop on Non-Monotonic Reasoning*, 337–343, October 17-19, 1984.

[17] J. Minker and A. Rajasekar. A Fixpoint Semantics for Disjunctive Logic Programs. To appear in Journal of Logic Programming.

[18] J. Minker and A. Rajasekar. Procedural Interpretation of Non-Horn Logic Programs. In E. Lusk and R. Overbeek, editors, *Proc. 9th International Conference on Automated Deduction*, pages 278–293, Argonne, IL, May 23-26, 1988.

[19] R. C. Moore. Semantical Considerations on Nonmonotonic Logic. *Artificial Intelligence*, 25(1):75–94, 1985.

[20] T. Przymusinski. On the Relationship Between Logic Programming and Non-Monotonic Reasoning. In *Proc. AAAI-88*, pages 444–448, 1988.

[21] T.C. Przymusinski. On the declarative semantics of deductive databases and logic programming. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann Pub., Washington, D.C., 1988.

[22] A. Rajasekar, J. Lobo, and J. Minker. Weak Generalized Closed World Assumption. To appear in Journal of Automated Reasoning.

[23] A. Rajasekar and J. Minker. *On Stratified Disjunctive Programs*. Technical Report CS-TR-2168 UMIACS-TR-88-99, Department of Computer Science University of Maryland, College Park, December 1988.

[24] R. Reiter. Circumscription implies predicate completion (sometimes). *Proc. Amer. Assoc. for Art. Intell. National Conference*, 418–420, 1982.

[25] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13(1 and 2):81–132, April 1980.

[26] R. Reiter and G. Criscuolo. On Interacting Defaults. *Proc. IJCAI 7*, 270–276, 1981.

[27] M.H. van Emden and R.A. Kowalski. The Semantics of Predicate Logic as a Programming Language. *J.ACM*, 23(4):733–742, 1976.

[28] 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 1149–176, Morgan Kaufmann, 1988.

Modelling Topological and Metrical Properties in Physical Processes

D.A. Randell and **A.G. Cohn**
Dept of Computer Science
University of Warwick, Coventry
CV4 7AL England
dr@cs.warwick.ac.uk agc@cs.warwick.ac.uk

Abstract

Developing suitable representations for formalising non-trivial domain knowledge has always been central to AI. Within Naive Physics ie. the task of encoding experiential knowledge of the world, few formal theories have appeared that exhibit formal elegance, conciseness and generality to cover a wide variety of modelling problems. We outline a first order formalism being developed that meets these criteria. The formalism is particularly attractive in that it provides the user with the means to model either spatial and/or temporal information as required. The power of the formalism is illustrated by modelling the process of phagocytosis of the amoeba, together with an outline of how many properties of physical entities and relations between them can be modelled within a unitary framework.

1.0 Introduction

The importance of representation within a formal framework has always been a central topic for discussion within AI. This has been particularly noticeable since Hayes' [1979, 1985a] call for the development of formalisms supporting a clean semantics which can be used for the representation of and reasoning about entities in non-trivial domains. In the spirit of Hayes' Naive Physics programme, we outline a concise rigorous formalism currently being developed. This exploits both topological and metrical information inherent in descriptions and explanations encountered in everyday discourse about the world. The formalism covers the representation of spatial and temporal topological information, and metrical information. Noting the controversy Hayes' programme seems to have engendered [Levesque, 1987] we are encouraged that this approach actually bodes well for at least some of the central ideas Hayes advocated.

The main structure of the paper is as follows. Section 2 discusses the need to capture topological and metrical information, and relates this to assumptions underlying this and comparable work. Section 3 is a brief overview of related work. Section 4 covers the formalism itself. The next two sections outline how empirical information is added and exploited and how processes are generated. Finally we illustrate the use of the formalism with a description of phagocytosis.

The financial support of the SERC is gratefully acknowledged.

2.0 Motivation

Barr [1965], points out that much informal discourse about the everyday world often exploits topological rather than geometrical information. This often happens on an unconscious level but it takes little reflection to see that this is indeed so. Most descriptions used to inform where things are, are topological and metrical rather than geometrical in nature. For example we may say that something is inside or can go inside something else; or that in order to get to some place one must go around some object, or pass over, under or indeed through it. That geometrical information is used in some everyday activity is not denied, what we do claim is that in ordinary circumstances its use is avoided. Since the understanding and representation of space and time is so fundamental and underpins much human activity, the construction of a formalism that can capture sufficient topological and metrical information yet support some interesting deductions would seem to be a central task in AI.

While we acknowledge that the task of capturing this very general, experiential knowledge of the world has much in common with previous attempts to model common sense knowledge and qualitative geometry, it is important to point out some differences that exist in the assumptions we have adopted and those which seem to underpin comparable work. First of all we do not assume that entities posited in some description or theory of the world are necessarily those that are exploited by the brain giving rise to such appropriate behaviour on our part. While it is true that any theory of the world must posit a set of entities, one cannot simply assume that entities posited in a naive theory are appropriate candidates to be introduced into some formal theory. This is to pay too

much attention to the notion of representation and to pay too little attention to the problem of primitive reference, ie. causal (theories of) reference.

Primitive reference is distinct from reference in the Fregean sense. Reference applies to the truth valuation for some proposition, but this assumes primitive reference being the means by which a name actually gets linked to some object before that truth valuation is given. This is not provided by Tarskian semantics, but is assumed. Only when primitive reference is better understood can we begin to make the stronger claim that the choice of entities in a theory have correlates in cognitive functioning and in common sense activity.

With respect to common sense reasoning in a formal framework, choice of an appropriate ontology is easily overlooked – particularly when introspection seems to provide an 'obvious' set of entities. Hayes [1979] makes the point that without an appropriate ontology the formalism becomes either too complex or too weak for the modelling problem at hand. But given that simple descriptions and explanations of and about the world (including attempts to explain problem solving) do not necessarily furnish us with the 'right' ontology (or set of purported rules underlying our decisions), we are at liberty to use any ontology we like, as long as the end result is capable of being measured against some agreed set of criteria by which theories, representations etc. can be compared. In this respect we do not pay too much attention to naive descriptions of the world supposedly giving us an appropriate ontology for a theory of the world, nor do we assume there is a substratum of 'deep knowledge' that is worth garnering. Such emphasis indicates an assumption that whatever underlies commonsense knowledge and activity has clear propositional content (which seems to ignore the nature and role of perception). This is related to another troublesome assumption pervading much AI – that cognition can be adequately modelled in a solipsistic environment.

Hayes [1979, 1985a] argues for a prolix ontology within Naive Physics. He also argues that such formalisms should have good conceptual cover (ie. having enough concepts to express what one wants to express), yet be capable of supporting dense inferential links between the formalism's concept tokens. The latter motivation stems from model theory, where undesirable rival models providing an interpretation of an axiomatisation are systematically eliminated. Strictly speaking this practice cannot guarantee a reduction of the number of supporting models. Any interesting first order axiomatisation will support a denumerable number of models, whether broad and dense or not, hence it remains an open question to what extent the idea of initially assuming a prolix ontology will help matters.

The primitives and axioms in our formalism have been kept to a minimum although formal independence of these has not been established. Although often associated with the domain of the logician, this practice has uses within AI and the cognitive disciplines; for given the deductive paradigm, a formalised theory with its primitives, definitions and independent axioms can help cast light upon a minimum set of entities and conditions required by a theory, the role of inference and its connection with cognitive functioning.

Although the deductive paradigm has its critics and limitations eg. see Levesque [1987], if theories concerned with our experience of the everyday world can be given a formal axiomatic treatment, we claim that topological and metrical information must play a significant part. To reiterate an earlier point, we do not assume the entities used to interpret our formalism are those that actually underpin perception and commonsense activity. However, it is interesting to note that many of the abstract concepts are, with a little reflection, surprisingly intuitive.

3.0 Related Work

Hayes [1979, 1985a,b] and Welham and Hayes [1984] discuss the importance of building large scale formalisms capable of encoding topological and metrical information. Hayes' [1985b] paper still remains his main contribution to applied Naive Physics. In this paper Hayes starts to develop a theory of topology but hits many problems, eg. capturing the physical relation of 'touching' objects, and whether or not a face of an object should be treated as a part of that object. In Welham and Hayes [1984] an attempt is made to fix some of these problems, in particular how objects can 'touch'. Cunningham [1985] discusses further problems.

A significant collection of papers dealing with commonsense knowledge and reasoning appears in Hobbs and Moore [1985] and Hobbs et al [1985]. The latter includes work by Kautz who discusses a formalism for the representation of spatial descriptions and concepts. Hager tackles the representation of properties of materials, and Shoham the representation of kinematics and shape. Many of these papers suffer to some extent from a free use of "axioms" without demonstrating the worth of the respective formalisms in terms of interesting deductions. An alternative approach arises with Leyton's [1988] process grammar which exploits curvature extreema to infer basic processes. Transformation of shape is covered but the modelling is restricted to capturing the geometrical properties of individual entities. Allen [1981, 1983] and Allen and Hayes [1985, 1987] develop a theory of time that exploits topological information, but a unitary formalism that can capture both spatial and temporal topological information is not considered.

Outside AI proper, Woodger [1939] makes a significant attempt to formalise an empirical science. He develops a formalism including the mereological[1] part/whole relation and others, to capture many concepts central to biology. Mereology also arises in Laguna [1922], Whitehead [1978] and Tarski's [1956] attempt to formalise geometry starting with an ontology of solids, rather than points. Carnap's [1958] axiom systems, provide a good hunting ground for any one interested in building deductive theories.

4.0 The Formalism

The formalism is expressed in a many sorted logic (msl) that allows arbitrary ad hoc polymorphism on predicates and functions. Sort restrictions on variables derive implicitly from the argument positions they occur in. The logic LLAMA [Cohn, 1987] satisfies these requirements. Space does not allow a detailed presentation of the logic or the sort structure and the sort declarations for all of the non logical symbols we shall employ. We shall briefly indicate the important sortal restrictions in the text. Important sorts are REGION, POINT, NULL and MEASURING SCALE (all mutually disjoint).

The formalism is rooted in a calculus of individuals developed by Clarke [1981, 1985]. Clarke utilises most of the mereological definitions that appear in Whitehead [1978] but from there on his work differs in several important respects. First Whitehead does not formalise his theory as does Clarke. The "quasi-Boolean", and "quasi-topological" operators and predicates used by Clarke are extensions of Whitehead's system. Finally Whitehead's assumption that individuals are continuous is dropped by Clarke.

Our formalism differs from Clarke's in the following respects. In the first instance Clarke uses second and even third order variables, while we keep our formalism strictly first order. We also add many new relations and functions to Clarke's existing set. Partial functions implicit in Clarke are made explicit in a msl, and a null object is introduced making them total on their domains. The use of continuity constraints to link intervals over which some particular property remains constant is new, as is the use of the calculus for the modelling of basic topological and metrical processes of the everyday world. Clarke presents his system as an 'uninterpreted' calculus.

We assume a domain of discourse that has as its basic set of primitive entities, (spatial and/or temporal) regions and points, numbers, measures, and a series of constants of measure. Informally, the basic regions may be thought to be potentially infinite in number with no restriction as to the degree of overlapping allowed. Each region coincides with a set of incident points, and every region is contained in a distinguished region called the universe.

Two primitive relations are initially introduced: 'C(x,y)' read as 'x connects with y' and 'B(x,y)' as 'x is wholly before y'. In order to help develop and guide the desired intuition needed to understand this formalism, we follow Clarke's example by providing intuitive interpretations for a sufficient number of relations.

In terms of points incident in regions, C(x,y) holds when two regions connect; of the incident points contained in both regions, at least one incident point is shared. Similarly B(x,y) holds when one of two regions are wholly before the other (in time) ie. all the points incident in one region are also wholly before the points incident in the other.

We will talk about regions being open and closed. These are topological concepts. A closed region is one that contains all its boundary points (more correctly all its limit points), whereas an object is open if it has no boundary points at all. Although we only explicitly refer to open and closed regions, it is important to point out that being open and being closed are not mutually exhaustive properties.

C(x,y) covers all cases of connection between regions from external contact (or 'touch') to all instances of mutual penetration including mutual total overlap or identity. Figure 1 illustrates the intended meaning of C(x,y) with pairs of (topologically closed) regions that satisify the relation. B(x,y) covers all cases where x is temporally before y though excluding the case of temporal abutment.

A set of axioms is given that govern C(x,y) and B(x,y):

$$\forall x[C(x,x)\wedge\forall y[C(x,y)\rightarrow C(y,x)]]$$
$$\forall xy[\forall z[C(z,x)\leftrightarrow C(z,y)]\rightarrow x=y]$$
$$\forall x[\neg B(x,x)\wedge\forall y[[B(x,y)\wedge B(y,z)]\rightarrow B(x,z)]]$$
$$\forall xy[B(x,y)\rightarrow [\neg C(x,y)\wedge\forall zw[[P(z,x)\wedge P(w,y)]\rightarrow B(z,w)]]]$$

C(x,y), is totally reflexive and symmetrical. B(x,y) is irreflexive, asymmetrical and transitive. The relation P(x,y) is dealt with below.

The basic set of merelogical relations defined in terms of C(x,y) are given as follows: 'DC(x,y)' is read as 'x is disconnected from y'; 'P(x,y)' as 'x is a part of y'; 'PP(x,y)' as 'x is a proper part of y'; 'O(x,y)' as 'x overlaps y' and 'DR(x,y)' as 'x is discrete from y':

$$DC(x,y) \equiv_{def.} \neg C(x,y)$$
$$P(x,y) \equiv_{def.} \forall z[C(z,x)\rightarrow C(z,y)]$$
$$PP(x,y) \equiv_{def.} P(x,y)\wedge\neg P(y,x)$$
$$O(x,y) \equiv_{def.} \exists z[P(z,x)\wedge P(z,y)]$$
$$DR(x,y) \equiv_{def.} \neg O(x,y)$$

DC(x,y) is understood to mean that regions x and y share no incident point in common; P(x,y) when all the points incident in x are also incident points in y; PP(x,y) when

[1] "Mereological" comes from a Greek root, meaning part.

all the points incident in x are also incident points in y, but not vice versa; O(x,y) when x and y share at least one common interior point; and DR(x,y) when either x and y share no point in common or when x and y share a point in common but share no interior points (ie. when x and y share only boundary points).

DC(x,y) implies DR(x,y) but not vice-versa: two regions may be discrete yet can be disconnected or in external contact (see Figures 1 and 2). It is also worth emphasizing here that the overlap relation O(x,y) does not capture the physical relation of covering, neither in the case of surface contact between two objects or optically, as in the case when one object occludes another; rather, the intended meaning of overlap is one of varying degrees of mutual penetration between regions.

Similarly, care is needed with the intuitive understanding of the part/whole relations since not all nuances associated with the concept of something being a part of something else are necessarily captured with this relation. In the case of an amoeba which surrounds and completely engulfs some food source, for that foodsource to be considered as a part of the amoeba in the sense captured by our formalism, the relation of the foodsource to the amoeba would be identical to that between the amoebal protoplasm and the amoeba as a whole. If on the other hand we wish to describe the foodsource as being inside the amoeba but forming no part of the amoeba itself, then the relation P(x,y) cannot be used to capture this notion of containment. For this additional formal machinery must be used. This is dealt with in more detail below.

The axioms imply that DC(x,y) is irreflexive and symmetrical, P(x,y) is totally reflexive, antisymmetric and transitive, PP(x,y) is irreflexive, asymmetrical and transitive, O(x,y) is totally reflexive and symmetrical, and DR(x,y) irreflexive and symmetrical.

The distinction between C(x,y) and O(x,y) provides the basis for a set of relations that are not commonly associated with calculi of individuals, yet are central for capturing a set of desired topological properties. 'EC(x,y)' is read as 'x is externally connected to y', 'TP(x,y)' read as 'x is a tangential part of y' and 'NTP(x,y)' read as 'x is a nontangential part of y' are now introduced:

$$\begin{aligned} &EC(x,y) \equiv_{def.} C(x,y) \wedge \neg O(x,y) \\ &TP(x,y) \equiv_{def.} P(x,y) \wedge \exists z[EC(z,x) \wedge EC(z,y)] \\ &NTP(x,y) \equiv_{def.} P(x,y) \wedge \neg \exists z[EC(z,x) \wedge EC(z,y)] \end{aligned}$$

EC(x,y) is to be understood when x and y share a point in common they share no interior points (ie. when only boundary points are shared), TP(x,y) when all the points in x are points of y and some other region z exists such that x,y and z share a common boundary point, and NTP(x,y) when all the points in x are points of y but there is no region z such that x,y and z share a common boundary point.

The following theorems arise: EC(x,y) is irreflexive and symmetrical, TP(x,y) is weakly reflexive and antisymmetric, and NTP(x,y) antisymmetric and transitive. A substantial list of theorems generated from these definitions and the above axioms can be found in Clarke [1981]. To compare Clarke's calculus with other calculi of individuals see Eberle [1970].

A set of configurations satisfying these (and other relations defined below) is given in Figure 1. For reasons of clarity we have assumed that the regions in question are all closed, but it must be emphasised that the formalism can accommodate both open and closed regions. We also assume each of the x,y paired regions to be surrounded by another region z acting as an environment. This 'environment' functions as the additional region z in external contact with x and y for TP(x,y) to hold, and only in external contact with y for NTP(x,y) to hold. The existence of this region is guaranteed by the function c-compl(x) (defined in section 4.8).

Figure 2 represents the basic lattice which orders the above relations in terms of their implicational strength. The weakest and most general relations are directly linked to $\top$ and the strongest directly linked to $\bot$ interpreted as tautology and contradiction respectively.

Twelve additional relations (eleven if identity is excluded) are added to Clarke's original set. These include the following: 'TPP(x,y)' read as 'x is a tangential proper part of y', 'NTPP(x,y)' as 'x is a nontangential proper part of y', 'PO(x,y)' as 'x partially overlaps y', 'x=y' as 'x is identical with y', 'TPI(x,y)' read as 'x is a tangential part of and identical with y' and 'NTPI(x,y)' as 'x is a nontangential part of and identical with y'; together with inverses of all the asymmetrical relations. Names and definitions for the inverse relations are not given in the interest of space.

$$\begin{aligned} &TPP(x,y) \equiv_{def.} TP(x,y) \wedge \neg P(y,x) \\ &NTPP(x,y) \equiv_{def.} NTP(x,y) \wedge \neg P(y,x) \\ &PO(x,y) \equiv_{def.} O(x,y) \wedge \neg P(x,y) \wedge \neg P(y,x) \\ &x=y \equiv_{def.} P(x,y) \wedge P(y,x) \\ &TPI(x,y) \equiv_{def.} TP(x,y) \wedge x=y \\ &NTPI(x,y) \equiv_{def.} NTP(x,y) \wedge x=y \end{aligned}$$

TPI(x,y) and NTPI(x,y) are simply added to the other relations to complete the lattice depicted in Figure 2.

At this point we drop the systematic practice of supplying intuitive interpretations of relations in terms of incident points, since the increased complexity of the linguistic descriptions begins to outweigh their intuitive usefulness.

TPP(x,y) is irreflexive and asymmetrical, NTPP(x,y) irreflexive, asymmetrical and transitive, PO(x,y) irreflexive and symmetrical, x=y totally reflexive, symmetrical and transitive, and TPI(x,y) and NTPI(x,y) weakly reflexive, symmetrical and transitive. The identity

relation x=y captures the non-logical notion of identity between regions in terms of mutually shared parts.

It is worthwhile pointing out the difference between the relations NTP(x,y) and NTPP(x,y). This may be confusing since they both share the same set of configurations depicted in Figure 1. A region is nontangential part of itself iff it has no object externally connected to it, ie:

$\forall x[NTP(x,x) \leftrightarrow \neg\exists y[EC(y,x)]]$

If NTP(x,x) is true then x is an open region. This is stated explicitly in the formalism. Because the set of configurations illustrated in Figure 1 are closed regions, NTP(x,x) cannot be represented. A boundary (or a least part boundary) of an object is required for some other object to externally connect with it. NTPP(x,y) should now be straightforward to understand. As it stands the above set of relations (excluding the relation B(x,y)) can support a purely spatial interpretation. It is worth keeping this interpretation in mind when considering the Boolean and topological operators.

4.1 The Boolean operators

The prefix "quasi-..." in "quasi-Boolean" (and "quasi-topological") is used by Clarke to distinguish his set of operators from those that arise in a complete Boolean (and closure) algebra. It is common place in calculi of individuals to refuse to postulate an individual that would be the analogue of the null element in a standard Boolean algebra. Clarke's calculus is no exception and supports no null object. By doing this, Clarke can only prove certain theorems hold when certain existential preconditions are satisfied, eg. that a complement exists for a given region only if it is not the universal region, and that an intersection of two regions exists only if those regions overlap. This complicates some of his proofs.

Unlike Clarke we introduce a null object as a constant into our domain allowing us to make all our functions total. A msl can cope with domain restrictions such as denying a complement to the universal region, but cannot deal with the fact that two disconnected regions have no intersection. While it is possible to use a Russellian theory of descriptions to eliminate functions contexually in terms of relations, identity and quantifiers (thereby resolving the problem of non-existence for certain values of functions), we choose to use pure functor notation. This is more compact and perspicuous than relational notation, and is more in keeping with the motivation to cut down the search space during mechanised inference. Ontologically speaking, introducing a null object becomes questionable[2] but its introduction is motivated by pragmatic convenience. In any case we could re-express the whole formalism using a Russellian analysis and thereby resolve at least some of the philosophical objections that may arise. Cohn [1989] shows that a msl allows the null object to be excluded from being a region (providing overlapping [Cohn 1987] is allowed).

We add Clarke's sum, complement,[3] Universe, and product operators. In addition we add the null object and difference operator. 'sum(x,y)' is read as 'the sum of x and y', 'compl(x)' is read as 'the complement of x', the constant 'U' is read as 'the universe', 'prod(x,y)' is read as 'the product (or intersection) of x and y', 'N' is read as 'the null object', and 'diff(x,y)' is read as 'the difference of x and y'. The iota operator 'ιx' is read as 'the unique x such that', but as stated before its use here does not necessarily indicate a Russellian analysis.[4]

$sum(x,y) =_{def.} \iota z[\forall w[C(w,z) \leftrightarrow [C(w,x) \vee C(w,y)]]]$
$compl(x) =_{def.} \iota y[\forall z[C(z,y) \leftrightarrow \neg P(z,x)]]$
$U =_{def.} \iota x[\forall y[C(y,x)]]$
$prod(x,y) =_{def.} \iota z[\forall w[C(w,z) \leftrightarrow [C(w,x) \wedge C(w,y)]]]$
$N =_{def.} \iota x[\forall yz[x=prod(y,z) \wedge DC(y,z)]]$
$diff(x,y) =_{def.} prod(x,compl(y))$

4.2 The topological operators

The representational power of this formalism is greatly increased with the introduction of topological operators. Clarke defines the interior, closure, exterior operators and the predicates Open and Closed. To this we add the boundary operator. 'int(x)' is read as 'the interior of x', 'cl(x)' is read as 'the closure of x', 'ext(x)' is read as 'the exterior of x', 'Open(x)' as 'x is open', 'Closed(x)' as 'x is closed'. and the operator 'bound(x)' is read as 'the boundary of x'.

$int(x) =_{def.} \iota y[\forall z[C(z,y) \leftrightarrow \exists w[NTP(w,x) \wedge C(z,w)]]]$
$cl(x) =_{def.}$
$\quad \iota y[\forall z[C(z,y) \leftrightarrow \exists w[\neg C(w,int(compl(x))) \wedge C(z,w)]]]$
$ext(x) =_{def.}$
$\quad \iota y[\forall z[C(z,y) \leftrightarrow \exists w[NTP(w,compl(x)) \wedge C(z,w)]]]$
$Open(x) \equiv_{def.} x=int(x)$
$Closed(x) \equiv_{def.} cl(x)=x$
$bound(x) =_{def.} prod(cl(x),cl(compl(x)))$

An axiom is added that guarantees that each region has an

[2] See Geach [1980] for a humorous yet instructive point that the temptation to treat "nothing" as a name opens the way to innumerable fallacies.

[3] A justification for defining complement as given and not as:
$compl(x) =_{def.} \iota y[\forall z[O(z,x) \leftrightarrow \neg P(z,y)]]$
as used by Goodman [1966], is that in terms of incident points, Clarke's operator guarantees that no point in the complement of an object is incident in the boundary of the object, whereas complement as defined by Goodman allows such a model. Since Clarke introduces points into his ontology, as do we, we choose Clarke's definition.

[4] When we write $\alpha(x) \equiv_{def.} \iota x[\Phi(x)]$ we mean $\forall y \Phi(y) \leftrightarrow y=x$

interior and that the intersection of two open regions is also open:

$$\forall x[\exists z[NTP(z,x)\wedge \forall y[\forall u[\neg EC(u,x)\wedge\neg EC(u,y)]\rightarrow \forall w[\neg EC(w,prod(x,y))]]]]$$

4.3 Atoms

An atom is a region that has no proper parts: the only part of an atom is itself. Every region contains an atom.

$$ATOM(x) \equiv_{def.} \forall y[P(y,x)\rightarrow y=x]$$
$$\forall x\exists y[ATOM(y)\wedge P(y,x)]$$

If the topological operators are excluded for the moment, atoms can either externally connect, be identical or be disconnected. With interiors defined over atomic regions, this no longer applies. By definition an atom has no parts other than itself, so if atoms are allowed to have interiors, the interior of an atomic region must be identical to that atom, hence atoms become open regions. Defining interiors over atomic regions produces an interesting deductive result, for it can be proved that when two non-atomic regions externally connect or 'touch', no two atoms within those regions 'touch'. This formal result casts some light on the naive conundrum of how if (physical) atoms are points with fields, (topologically open?), and atoms make up objects, how is it that objects can 'touch'? A similar conundrum arises if physical objects are construed as sets of points: how is it the two objects can touch when points can only be discrete or identical? The formalism supporting open atoms illustrates what may be seen as an informal fallacy at work, namely the fallacy of composition. This is the mistake to assume that all the properties of parts of a whole must belong to that whole.

4.4 Points

Many domains can be adequately modelled using only regions. However points are included in the formalism for the following reasons. Firstly the chosen intuitive interpretations for the mereological relations are vindicated with the introduction of points; secondly, adding points allows an interesting comparison to be made with extant formalisms (eg. Allen and Hayes' [1987] formalism supports points); and thirdly, the descriptive power of the formalism increases, eg. enabling the relation TPP(x,y) to be restricted so that only x,y and z connect at a single boundary point (see section 4.8)

Clarke introduces points by using second order variables, whereas here only first order variables are assumed.

Initially we constructed points as we did for atoms, but this generated problems with the part/whole relation and the intuitive interpretation given for these mereological relations. In particular if two regions externally connect, sharing a point in common, and if a point is taken to be a part region, the two regions must overlap (by definition of overlap). But externally connected objects do not overlap (by definition)! Similar problems arise if faces are taken to be parts of objects, and highlights again a classic problem (eg. Hayes' [1985b] difficulties when treating faces as parts of objects and his subsequent introduction of the notion of a directed surface). In order to circumvent this, points are treated as primitive entities and linked to regions by means of a new relation.

'IN(x,y)' read as 'x is incident in y' is introduced, and restricted so that the extensions of its arguments cannot be of the same geometric dimension, nor is the second argument allowed to be of a lower dimension than the first. In contrast the part/whole relation is defined so that it can only take arguments whose extensions have the same dimension. Thus eg, while a face is allowed to have parts, the face is not a part of the object which has that face. But a face can be incident in that object. These sort restrictions imply that IN(x,y) is irreflexive and asymmetrical. The following axioms relate IN(x,y) to the mereological relations.

$$\forall xy[C(x,y)\leftrightarrow\exists z[IN(z,x)\wedge IN(z,y)]]$$
$$\forall xy[P(x,y)\leftrightarrow\forall z[IN(z,x)\rightarrow IN(z,y)]]$$

The following theorem arises:

$$\forall xy[\forall z[IN(z,x)\leftrightarrow IN(z,y)]\leftrightarrow x=y]$$

Identity of points/boundaries is stipulated with the axiom:

$$\forall xy[\forall z[IN(x,z)\leftrightarrow IN(y,z)]\leftrightarrow x=y]$$

When describing eg. plane figures embedded in 3-space, it is important to realise that any atoms posited in the plane are to be conceived as open discs and not open balls.

4.5 Scattered or fragmented vs connected regions

Mereology has been adopted and criticized as an attempt to deal with the problem of divided reference and providing an adequate treatment of mass terms extensions (see eg. Bunt [1985], and Pelletier [1979]).

With regard to physical bodies, mass terms relate to what could be broadly thought of as stuff eg. water, flour and sand. Count terms relate to things. Grammatically, mass and count terms are paired with the adjectives "less" and "fewer" respectively, and whether a term is regarded as one or the other in some sentence depends upon which adjective can be correctly applied.

Mereology allows scattered or connected ('one piece') objects. Individuation of physical objects often appeals to notions of connectivity or coherence. But there is no logical reason why this must be so. Traditional quantification theory based on set-theoretic foundations enforces this conception, since without the part/whole relation, identity and diversity are the only logical relations that can hold between objects. Although the notion of a scattered object might at first seem odd (except perhaps to the mathematician eg. the disjoint union of two sets) simple

everyday examples are easy to find. Consider a closed, pump and conduit system. We can envisage the fluid passing through a series of chambers in its passage around the system, yet if we were to view the space taken up by the fluid at particular points in time, we might well find a scattered object; yet we tend to talk of that fluid as a unitary body, eg. 'the fluid in the pump'.

A far more unintuitive result and damaging criticism of mereology arises from the unrestricted use of the sum operator, which allows the union of any two objects or bodies in the universe of discourse. Eberle [1970], points out that if we assume a domain of physical objects, then the sum of two distant stars can hardly be considered to be another individual. Sortal restrictions on the sum operator might help, eg. restricting scattered objects to STUFFS, but awkward cases exist eg. when referring to a broken cup scattered in bits all over the floor.

Clarke defines separated and connected regions as follows: 'SEP(x,y)' is read as 'x is separated from y', and 'CON(x)' as 'x is connected':

$$\boxed{\begin{array}{l} SEP(x,y) \equiv_{def.} DC(cl(x),y) \wedge DC(x,cl(y)) \\ CON(x) \equiv_{def.} \neg\exists yz[sum(y,z)=x \wedge SEP(y,z)] \end{array}}$$

The relation CON(x) denotes those regions that cannot be divided into two mutually exhaustive disjoint parts.

4.6 Hollow, toroidal, simply connected and multiply connected regions

Hollow regions are easily definable given disconnected regions. 'HOL(x)' read as 'x is hollow' is defined as:

$$\boxed{HOL(x) \equiv_{def.} \neg CON(compl(x))}$$

Topologically speaking in 2-space this would generate an annulus, in 3-space a hollow ball as distinct to solid (or filled) ball. Note that we do not call a hollow ball in 3-space a sphere. A sphere is characterised as a region that is surface only (see section 4.10).

Modelling multiply connected regions in 3-space, requires more formal work. Only one class of toroidal regions have disconnected complements – the prototypical bicycle tyre inner tube. The solid torus does not. We need some means to express the property of simple connectedness.

A simply connected object is one in which every closed loop incident in that object can be shrunk to a point within that object. If the object has a hole[5] it is clear that this operation cannot succeed – to shrink the loop would require it to pass through the boundary. As an everyday example of what we mean, a potter initially aims to produce a well worked lump of clay with no air pockets – such an object is simply connected. Subsequent pulling or compacting the clay will not alter this property, providing that the potter does not join any parts of the surface. We assume simple connectedness as a primitive property. 'SCON(x)' is read as 'x is simply connected'. We also assume toroidalness as a primitive property. 'MCON(x)' is read as 'x is multiply connected', 'TOR(x)' is read as 'x is a torus':

$$\boxed{\begin{array}{l} MCON(x) \equiv_{def.} CON(x) \wedge \neg SCON(x) \\ TOR(x) \rightarrow MCON(x) \end{array}}$$

Taken together with the convex hull operator described below, toroidal objects provide a useful means with which to start modelling filters, since a filter is just a multiply connected object with 'holes' of a particular size. Links of an ideal chain are also solid toroidal regions, while linked links of a chain simply have link parts passing 'through' another link's convex hull.

4.7 The convex-hull operator

In many cases we talk about objects being inside others without the contained object being part of the container. Thus as a simple example we often talk about water being inside a cup, or a bird in flight inside some aviary. To be inside does not necessarily mean the object in question is completely sealed off from sight. We can model this relation by using the notion of the convex-hull or convex-cover of a region. Informally, the convex-hull of a body is that region of space that would be defined if that object were wrapped in a taught 'cling-film' membrane, or completely sealed by an arbitrarily thin taught rubber membrane. In 2-space this would be akin to that region of space that would be described by a rubber band stretched to fit around some figure. The convex-hull operator can be applied to a set of points, or a set of disconnected regions. However, here we restrict the convex-hull operator to apply only to individual (one piece) objects. An object is then said to be inside another iff it is part of the convex-hull of that object and not part of that object. The addition of this function allows a richer description to be given with respect to either disconnected or externally connected regions. We introduce 'conv(x)', read as 'the convex hull of x' as a new primitive, give axioms to relate the function to the rest of the formalism, and define the relations 'Inside(x,y)' read as 'x is inside y', 'P-Inside(x,y)' read as 'x is partially inside y', 'Outside(x,y)' read as 'x is outside y', and 'W-Outside(x,y)' read as 'x is wholly outside y'.

[5] Strictly speaking the hole is a property of the surrounding space.

$\forall x SCON(conv(x))$
$\forall x P(x,conv(x))$
$\forall xy[C(x,y) \rightarrow C(x,conv(y))]$
$\forall x[conv(x)=conv(conv(x))]$
$Inside(x,y) \equiv_{def.} P(x,conv(y)) \wedge \neg P(x,y)$
$P\text{-}Inside(x,y) \equiv_{def.} PO(x,conv(y) \wedge \neg P(x,y)$
$Outside(x,y) \equiv_{def.} DR(x,conv(y) \wedge \neg P(x,y)$
$W\text{-}Outside(x,y) \equiv_{def.} DC(x,conv(y))$

4.8 'Surround' analogues of the part/whole relations

Consider the pictorial representations ascribed to the part/whole relations in Figure 1, as nested circles (and not discs). We could fill and describe them in at least two ways. In the first case we could fill the inner circle to make a region, and then fill the other circle so as to make the inner a part of the outer. But we could equally fill the configuration so that the inner is surrounded by an outer annulus or 'crescent' as a discrete region. In the latter case the two regions would not be related as part to whole, but one would surround the other.

Given a domain of closed regions we can view this new surround relation as an instance of external connectedness. Because containment being entertained here is an asymmetrical relation, we can see that it seems possible to define a 'surround' analogue of NTPP(x,y), since clearly the relation between x and y would be asymmetrical, ie. x relative to U would always be inside y. We would like to define a surround analogue of TPP(x,y), but this is impossible. This may seem surprising at first, but any difficulty in seeing this usually arises with the way we have chosen to represent the proper part relations pictorially. Since no metric is being assumed, TPP(x,y) is satisfied by all the configurations in Figure 3. Notice that with respect to the idea of a surround analogue, either region can be the surround of the other; hence the necessary condition of asymmetry fails, and the asymmetrical surround analogue of TPP(x,y) becomes indefinable. We call this property 'inversion'. In order to capture some notion of surround analogue of the proper part relations, we define a new relation $TPP_p(x,y)$ which restricts the connection between x, y (and external connection to z) to a single point.

A new function is defined: 'c-compl(x)' read as 'the closed complement of x':

$c\text{-}compl(x) =_{def.} compl(int(x))$

c-compl(x), when x is closed, returns a region that includes x's boundary.

We define 'NTS(z,x)' read as 'z is the nontangential surround of x' , and 'TS(x,y)' read as 'x is tangentially surrounded by y' as:

$NTS(x,y) \equiv_{def.} \exists z[NTPP(x,z) \wedge y=prod(c\text{-}compl(x),z)]$
$TS(x,y) \equiv_{def.} \exists z[TPP(x,z) \wedge y=prod(c\text{-}compl(x),z)]$

As argued above, a surround analogue of TPP(x,y) requires an additional condition to hold to stop the inversion of x and y. We therefore define a surround analogue of TPP(x,y) so that the x,y, and z satisfying TPP(x,y) are single point connected only. '$TPP_p(x,y)$' is read as 'x is a boundary point connected tangential proper part of y' and, '$TS_p(x,y)$' read as 'x is tangentially surrounded by y (at a point)' is defined as:

$TPP_p(x,y) \equiv_{def.} P(x,y) \wedge \exists z[[EC(z,x) \wedge EC(z,y)] \wedge \iota u[Point(u) \wedge IN(u,x) \wedge IN(u,y) \wedge IN(u,z)]$
$TS_p(x,y) \equiv_{def.} \exists z[TPP(x,z) \wedge y=prod(c\text{-}compl(x),z)] \wedge \iota u[Point(u) \wedge IN(u,x) \wedge IN(u,y) \wedge IN(u,z)]]$

Given the relation between NTPP(x,y) and NTS(x,y) we envisage the construction and use of a metalevel rewrite rule indicated by the arrow '$\Rightarrow$' (or '$\Leftrightarrow$' for bi-directionality), read as 'can be redescribed as' linking related wffs, eg. $NTPP(x,y) \Leftrightarrow NTS(x,y)$, which would allow two modes of description between x and y; either x as a part of y, or x surrounded by but not part of y. Bi-directionality is set up between $TS_p(x,y)$ and its part/whole analogue $TPP_p(x,y)$

4.9 Metric spaces

A measure of an object is a relation between that object, a number and unit of measure. Below we list a few sample functions used in the formalism: the volume of x 'vol(x)' takes a body as its argument and maps this to an ordered tuple; the first element of which is a member from the set of non-negative real numbers and the second a unit of measurement cubed, eg. cm^3.

The temperature of x, 'temp(x)' takes a body (or surface of a body) as its argument and maps this to an ordered tuple; the first element of which is a member from the set of reals (within some to be decided range), and the second a unit of measurement, eg.° C:

The distance function 'd(x,y)' read as 'the distance between x and y' is defined in terms of the diameter of the smallest sphere (described in section 4.10) that has x and y as incident points, or in the case of regions x and y, the diameter of the smallest sphere that externally connects them both. The relation between a point and a region is similarly defined. Thus there are three definitions for d(x,y) depending on the sorts of x and y.

$d(x,y) =_{def.} \iota z[Sphere(z) \wedge IN(x,z) \wedge IN(y,z) \wedge \forall w[Sphere(w) \wedge IN(x,z) \wedge IN(y,z)] \rightarrow z \leq w]$
$d(x,y) =_{def.} \iota z[Sphere(z) \wedge EC(z,x) \wedge EC(z,y) \wedge \forall w[Sphere(w) \wedge EC(w,x) \wedge EC(w,y)] \rightarrow z \leq w)]$
$d(x,y) =_{def.} \iota z[Sphere(z) \wedge Point(x) \wedge IN(x,z) \wedge EC(y,z) \wedge \forall w[Sphere(w) \wedge IN(x,w) \wedge IN(x,z)] \rightarrow z \leq w]$

As an example of how such functions can be linked to the extant formalism, we add the following sample axioms:

$\forall xy[P(x,y) \rightarrow vol(x) \leq vol(y)]$
$\forall xy[C(x,y) \leftrightarrow d(x,y)=\langle 0,\_ \rangle$

The distance operator satisifes a set of axioms defining a metric. We introduce a ternary relation 'N(x,y,z)', read as 'x is nearer to y than x is to z' (along the lines of van Benthem [1982]) and use this relation to define the relation 'E(x,y,z)' read as 'y is as near to x as it is to z':

$N(x,y,z) \equiv_{def.} d(x,y)<d(x,z)$
$E(x,y,z) \equiv_{def.} \neg N(x,y,z)\wedge\neg N(x,z,y)$

With appropriate sortal restrictions, additional relations and axioms from van Benthem can be introduced and exploited.

4.10 Nested balls and the inverse square law

Many physical phenomena can be modelled by an appeal to the inverse square law, eg. the variation of the amplitude of a radial wave propagating across the surface of a pond, or the drop in illumination on a surface as the distance between the light source and the surface increases. The basis for describing the geometry of such phenomena is rooted in the contruction of a set of uniformally nested balls each sharing the same centre. The relative distance between the 'shells' of this set of balls or between a set of concentric spheres (as a ball without its interior), can be exploited so as to provide a basis from which estimates of the intensity of some energy source radiating through the nest can be made. In order to set this up we first of all assume 'Ball(x)' read as 'x is a ball' as a primitive, define a closed ball, concentricity, and then define the nest. 'C-Ball(x,y)' is read as 'x is a closed ball', 'CP(x,y)' is read as 'x is a concentric part of y':

$C\text{-}Ball(x) \equiv_{def.} Ball(x)\wedge Closed(x)$
$CP(x,y) \equiv_{def.} C\text{-}Ball(x)\wedge C\text{-}Ball(y) \wedge P(x,y)\wedge$
$\exists zuv[DC(z,u)\wedge DC(u,v)\wedge DC(z,v)$
$EC(z,y)\wedge EC(u,y)\wedge EC(v,y)\wedge$
$E(x,z,u)\wedge E(x,u,v)]$
$\forall x[C\text{-}Ball(x)\rightarrow\exists y[C\text{-}Ball(y)\wedge\neg(x=y)\wedge CP(x,y)]$

The definition of a sphere is simply defined as the difference between a ball and its interior:

$Sphere(x) \equiv_{def.} \exists y[C\text{-}Ball(y)\wedge x=diff(y,int(y))]$

5.0 Adding and exploiting empirical information.

Empirical information is added to the basic formalism by means of a series of axioms that link eg. physical properties of bodies and processes, to the relations, functions, and linked continuity constraints (described below). For example, gaseous bodies (not having any well defined boundary) share some of the formal characteristics of open regions, and like them can be contained. Wet bodies (cf. Hayes' [1985b] complex ontology) can be simply modelled by exploiting the interior function and predicating wetness to some object x and denying this with regard to its interior. A sponge may be viewed as the type of object that having a wet surface, has a wet interior (or at least soon will have!). Of course this is to view the sponge as a body without interstices; but given the notion of containment, and of multiply connected bodies, we can also adopt a finer level of abstraction and view the sponge as being full of interstices that contain the water, while the water does not form part of the sponge. Similarly, filtration may be modelled by stating that of two substances and a filter, the one may overlap the filter, while the other may not; or by introducing the notion of a multiply connected object, and impose a metric on the filter's holes, and the 'atomic' parts of the two to be filtered substances. Properties such as shear are typically linked to the change in size of the convex hull of an object over time, with no change in volume; and being compressed to a change in both. Space does not allow an empirical formalism along the above lines to be given, however we believe the above is sufficient to see how this would proceed.

6.0 Modelling Processes

We characterise states of affairs and events along the lines of Galton [1984]. States of affairs are viewed as descriptions of the world that are dissective (ie. typically remaining true over their sub-periods), obtain (at moments), are measurable, can be negated and are homogeneous; while events are unitary, occur (in intervals), have individual occurrences, can be counted, have no negation, and have distinct phases. Thus eg. a description that preserves some degree of topological continuity over a period of time can be seen as a state (of affairs), while two linked states with a change in the topological description would be regarded as an event. Processes are then regarded as a specified sequence of linked state changes.

The table in Figure 5 illustrates permissible topological transformations between pairs of objects obeying certain of the mereological relationships. The arrow '$\rightarrow$' indicates the direction of transformation allowed; the arrow '$\Rightarrow$' indicates possible re-description. The full table is not given here for lack of space, but it has a large number of entries which are blank. If only open objects are considered the reduction in the search space is particularly dramatic. Addition of taxonomic information, together with that extracted from the empirical part of the axiomatisation, also serves to cut down the number of possible transformations that two objects can make from one state to another. This is covered in more detail below.

7.0 A sample modelling problem: phagocytosis (and exocytosis)

The descriptive power of this formalism can be appreciated with the following challenge problem: how to model the process whereby one object engulfs another and then discharges that or another object. A familiar example of this arises in phagocytosis and exocytosis, the processes

used by cells to approach, surround and engulf particles, digest them, and then expel the waste material. The amoeba is often used to illustrate this behaviour, and is the organism that we adopt in our model.

Figure 4 represents a part sequence of the stages an amoeba passes through during phagocytosis. The amoeba approaches a food particle and then projects its pseudopodia to eventually wrap around and engulf the particle. In so doing a vacuole or fluid filled space (containing the liquid from its immediate environment) is formed, with the food particle as part of it. Enzymes within the amoeba are directed to the vacuole and enter into the vacuole. These break down the food into nutrient and waste. The nutrient is absorbed into the protoplasm leaving the waste in the vacuole. The waste is eventually discarded by a similar but inverse process as the amoeba moves forward.

Space does not allow a complete description of these processes, although it should be reasonably clear how the linguistic descriptions can be transformed into formal ones. Each stage is accompanied with its formal description, together with numbered add lists, and delete lists (signified by "– n").

Processes such as phagocytosis and exocytosis are named and defined by ordering a series of generalised state descriptions. The temporal relationship between such descriptions can be either assumed (as has been done here) or made explicit by introducing the temporal relations covered by our formalism. It is then a matter of choice how one wishes to use this information. The "state and state transformation" method of Green outlined in Chang and Lee [1973] can be used within our formalism to find a sequence of actions that will achieve a particular goal. In this case a state is understood to coincide with a particular description where the relation between the various elements of the configuration preserve some sense of (in this case) topological invariance. Actions are then construed as functions on states that indicate that one state can be transformed into another.

The use of continuity-constraint information can follow a similar pattern of use with the search space further reduced by the addition of empirical information about the domain. For example if we know that our objects are solid, rigid bodies, we know that these bodies cannot overlap, and as such the only transformations possible are external connection to being disconnected and vice versa; if one is the nontangential surround of the other no transformation can arise! However if one object is deformable and the other not, then many entries of the continuity constraint table can be cut from the search space. It is also interesting to note that the topological operators allow different 'environments' of objects under discussion to be separated out. This feature seems particularly useful if we wish to concentrate upon eg. some process that only arises within the amoeba. Sortal restrictions can also be exploited so that much 'useless' domain information need not clutter the search space.

8.0 Work continuing and further work

Using B(x,y) as our primitive, the 13 dyadic relations common to interval logics (eg. Allen and Hayes [1985, 1987] have been named and defined. We also recognise a temporal analogue for a disconnected spatial region available within the formalism, and its use when modelling activities that are eg. punctuated by periods of rest.

Additional modelling problems that have been tackled with this formalism include the operations of a simple pump, and the process of filtration. We hope the challenge problem of modelling the 'swooshing machine' [Hobbs et al, 1985] will be tractable, although this would require additional work capturing eg. some notion of causation, pressure and gravity.

On the computational side, independence of axioms within a large scale formalism suggests an interesting area of research. For example we have found the addition of lemmas for transitivity substantially reduced the search space of a standard resolution based theorem prover. The use of a transitivity table along the lines of Allen [1983] is envisaged.

9.0 Conclusions

Hayes [1979, 1985a] regarded the attempt to build large-scale formalisms as a useful test bed for exploring naive physics. In addition he argued that problems of search, the efficient management of inference in complex domains, together with pressures to produce working programs, were in part responsible for a certain lack of general extendability and usefulness of AI programs. He suggested control of useless inference (with respect to some goal state) might be achieved by adding metalevel information to the deductive interpreter, and anticipating a rich structure embedded in a naive physical formalism, thought this might be exploited towards such ends. In this respect we see our attempt (in the spirit of Hayes' programme), as a useful benchmark to explore his ideas, and within which the pragmatic elements of using such large scale formalisms can be assessed, naive physical or not.

Although the formalism is incomplete, and as yet we do not have a complete theory to implement, we are encouraged to see many indications of how unnecessary inference can be controlled by adopting the approach we have outlined, apparently vindicating Hayes' expectations. We are confident that the underlying richness of the concepts captured in this formalism makes it especially attractive as a general representation language where either static, ie. spatial, or spatial/temporal infor-

mation is in abundance.

Acknowledgements

We have benefited from many discussions with too many people to list here. However we are especially indebted to Tony Halbert and Roger Teichmann for their philosophical queries, and to Felix Hovsepian, Guy Saward and Ian Gent for their critical formal insights. We also acknowledge the comments of the anonymous referees who assessed the extended abstract of this paper; which in part led to the revision of our ontology mentioned in the text. The authors of this paper are indebted to their wives (Sarah Randell and Gillian Cohn) for their patience and encouragement during the preparation of this paper.

References

Allen, J. F. [1981], 'A General Model of Action and Time', Tech. Report 97, Dept Comp. Sci., University of Rochester.

Allen, J.F. [1983], 'Maintaining knowledge about temporal intervals', *CACM* 26(11), 832-843.

Allen, J. and Hayes, P.J. [1985], 'A Common Sense Theory of Time', *Proc. IJCAI-85*, Los Angeles, California, 528-531.

Allen, J. F. and Hayes, P. J. [1987], 'Moments and points in an interval-based temporal logic', Tech. Report 180, Dept. Comp. Sci. and Phil., University of Rochester.

Barr, S [1964], *Experiments in Topology*, John Murray, London, 154.

van Benthem, J. F. A. K. [1983], *The Logic of Time*, Synthese Library, Reidel, London, Appendix A.

Bunt, H. [1985], 'The Formal Representation of (Quasi-) Continuous Concepts', in Hobbs and Moore [1985], 37-70.

Carnap, R. [1958], *Introduction to Symbolic Logic and its Applications*, translated by H. Meyer and J. Wilkinson, Dover, New York, Chapters E to H and Appendix V.

Chang, C-L and Lee, R. C-T [1987], *Symbolic Logic and Mechanical Theorem Proving*, Computer Science Classics, Academic Press, London.

Clarke, B. L. [1981], 'A calculus of individuals based on 'connection'', *Notre Dame Journal of Formal Logic* 22(3), 204-218.

Clarke, B. L. [1985], 'Individuals and Points', *Notre Dame Journal of Formal Logic*, 26(1), 61-75.

Cohn, A. G. [1987], 'A More Expressive Formulation of Many Sorted Logic', *J. of Automated Reasoning* (3), 113-200.

Cohn, A. G. [1989], 'On the appearance of sort literals: a non-substitutional framework for hybrid reasoning, in *Principles of Representational Reasoning* ed. R. J. Brachman, H. Levesque and R. Reiter, Morgan Kaufmann, Los Altos.

Cunningham, J. [1985], 'A Methodology and a Tool for the Formalisation and representation of "common sense" (Naive Physical knowledge)', PhD thesis, University of Essex.

Eberle, R. A. [1970], *Nominalistic Systems*, Synthese Library, Reidel, Dordrecht, 40-41.

Galton, A. [1984], *The Logic of Aspect*, Clarendon Press, Oxford.

Geach, P. [1980], *Reference and Generality*, Cornell Univ. Press, Ithaca.

Goodman, N. [1966], *The Structure of Appearance*, Second Edition, Bobbs Merril.

Hayes, P.J. [1979] 'The naive physics manifesto' in *Expert Systems in the Micro Electronic Age*, ed. D. Michie, Edinburgh University Press.

Hayes, P.J. [1985a] 'The second naive physics manifesto', in Hobbs and Moore [1985].

Hayes, P.J. [1985b] 'Ontology for Liquids', in Hobbs and Moore [1985].

Hobbs, J. R. et al. [1985], 'Commonsense Summer: Final Report', Center for the Study of Language and Information, Report No. CSLI-85-35.

Hobbs, J. R. and Moore, R. C. [1985], *Formal Theories of the Common Sense World*, Ablex, Norwood, New Jersey.

de Laguna, T. [1922], 'Point, line, and surface, as sets of solids', *The Journal of Philosophy* 19(17), 449-461.

Levesque, H. ed. [1987], 'Taking Issue/Forum: A Critique of Pure Reason', *Comp. Intel.* 3(3), 149-237.

Leyton M, [1988], 'A Process Grammar for Shape', *Artificial Intelligence* 34, 213-247.

Pelletier, F. J. [1974], 'On Some Proposals for the Semantics of Mass Nouns', *J. of Philosophical Logic*, 3, 87-108.

Tarski, A. [1956] 'Foundations of the geometry of solids' in *Logic, Semantics*, Metamathematics, trans. J. H. Woodger, Oxford University Press, Oxford.

Woodger, J.H. [1937] *The Axiomatic Method in Biology*, Cambridge University Press, Cambridge.

Whitehead, A.N. [1978] *Process and Reality: Corrected Edition*, eds. D.R. Griffin and D.W. Sherburne, The Free Press, Macmillan Pub. Co., New York.

Welham, B. and Hayes, P. J. [1984], 'Notes on Naive Geometry', HP Labs Bristol and Schlumberger Palo Alto Research Labs.

Figure 1.

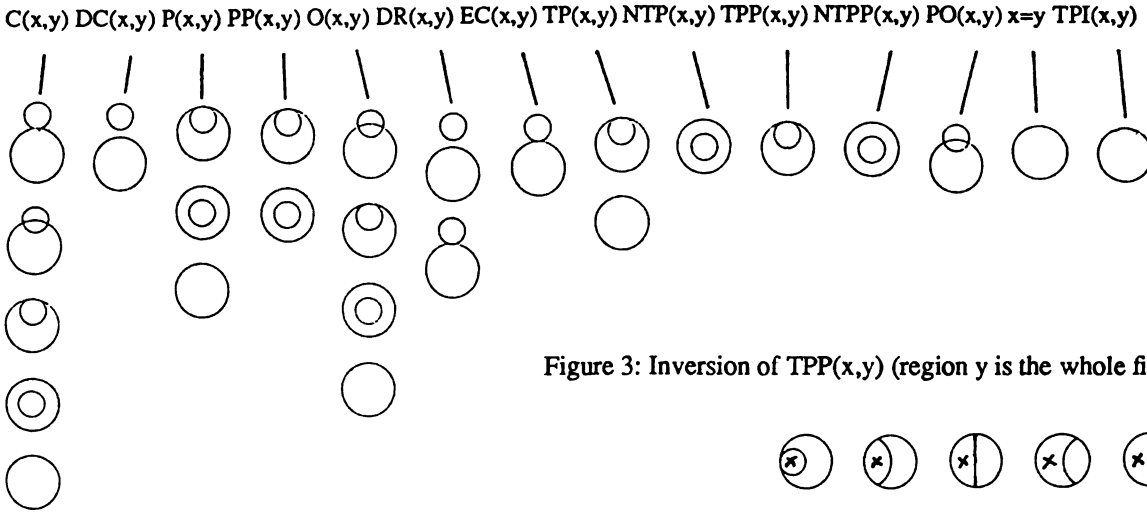


Figure 2: The basic mereological lattice

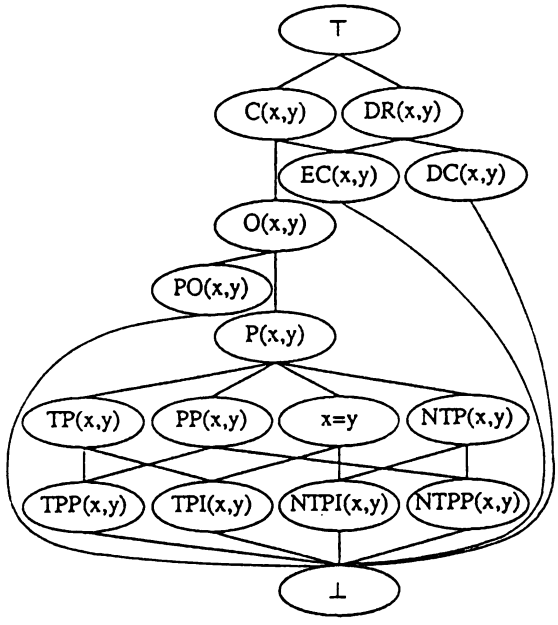
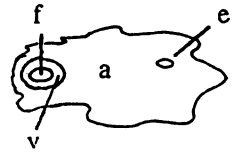


Figure 3: Inversion of TPP(x,y) (region y is the whole figure)

Figure 4: Phagocytosis

a=amoeba
e=enzyme
f=food
v=vacuole



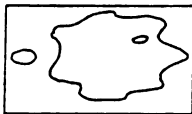
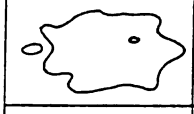
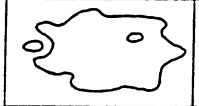
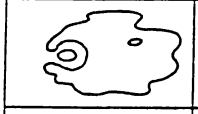
	1) W-Outside(f,a) 2) NTPP(e,a)	
	3) Outside(f,a) 4) DC(f,a) -1	
	5) P-Inside(f,a) -3	
	6) Inside(f,a) -5	
	7) NTPP(f,v) 8) DC(e,v) 9) TS_p(v,a)	7') NTPP(f,v) 8') DC(e,v) 9') TPP_p(v,a) -4,-6
	10) NTS(v,a) -9	10') NTPP(v,a) -9'
	11) EC(e,v) -8	11') EC(e,v) -8'

Figure 5: Part of the continuity/redescription table.

	DC	EC	PO	=
DC		→		
EC	→		→	
PO		→		→
=				

	TS_p	TPP_p	NTS	NTPP
TS_p		⇒	→	
TPP_p	⇒			→
NTS	→			⇒
NTPP		→	⇒	

Formal Theories of Belief Revision

Anand S. Rao
Australian AI Institute
Carlton, Vic-3053
Australia
Email: anand@aaii.oz.au

Norman Y. Foo
University of Sydney
Sydney, NSW-2006
Australia
Email: norman@basser.oz.au

Abstract

Belief revision has been an active area of research in AI and philosophy in recent years. This paper discusses two philosophical theories of belief revision, namely the *coherence* and *foundational* theories of belief revision. Unlike most philosophical works which treat coherence theory and foundational theory as two opposing theories, we view the foundational theory as an extension of the coherence theory of belief revision. Coherence theory is based on the intuitive principles of *minimal change* and *maximal coherence* and foundational theory is based on the above two principles and the *foundational thesis* which states that all beliefs are *justified* beliefs. By providing a possible-worlds semantics, axiomatization, and soundness and completeness results for both the theories, the paper presents a detailed and formal account of belief revision, lacking in most of the earlier work in this area.

1 Introduction

Belief revision is the process by which an agent revises her set of beliefs at the current instant of time, based on some input from the external world, to move into the next instant of time, possibly with a different set of beliefs. Belief Revision has been an active area of research in AI in recent years [Doyle, 1979, de Kleer, 1986, Martins and Shapiro, 1988]. Apart from researchers in AI, philosophers have also shown an active interest in this area [Harman, 1986, Alchourron *et al.*, 1985, Gardenfors, 1988]. The philosophical theories of belief revision(BR) fall into two broad categories - *coherence theory of belief revision(CTBR)* and *foundational theory of belief revision(FTBR)*.

The *foundational theory* of beliefs views each belief as either being self-evident or having a non-circular, finite sequence of justifications which terminate in a self-evident belief. FTBR [Harman, 1986] consists of subtracting any of one's beliefs that do not have a satisfactory justification, or adding new beliefs that either need no justification or are justified on the basis of other justified beliefs one has.

In *coherence theory* what justifies a belief is not that it is self-evident, nor that it has been deductively proved from a self-evident proposition, but that it *coheres* with a comprehensive system of beliefs. CTBR [Harman, 1986] states that one makes *minimal changes* to one's belief in order to maintain *coherence* or eliminate incoherence. Incoherency is often caused by explicit inconsistency. Thus the two main aims of CTBR are to *maximize coherence* and *minimize changes*.

A couple of examples will clearly illustrate the difference between the two theories. Consider a simple database with the following facts:

platypus(platty).
platypus(platty) $\rightarrow$ lives(platty,oz).

By consequential closure the database will also have the belief lives(platty,oz). Now if one removes the fact platypus(platty) from the database, the belief that lives(platty, oz) is no longer justified. FTBR insists that the unjustified belief lives(platty, oz) should be removed, while CTBR requires minimal change which is obtained by removing platypus(platty) but continuing to believe that lives(platty, oz). In this situation one might argue that FTBR provides the more intuitive answer. However consider the following example where CTBR gives the more inuitive answer.

The example given below is a simplified case of the Reaction Control System (RCS) of the space shuttle built using the Procedural Reasoning System (PRS [Georgeff and Lansky, 1986]). Assume that the database consists of the following facts:

pressure(he-tank, 400psi).
pressure(he-tank, X) $\wedge$ greater-than(X, 300psi)
$\rightarrow$ valve(he-tank, closed).

A high pressure (> 300psi) in the helium tank will trigger an action which will result in the belief valve(he-tank, closed). Now if the pressure drops below 300psi the belief valve(he-tank, closed) is no longer justified. However one cannot stop believing it, because the helium tank is still closed and will remain closed until an explicit action to open it is carried out. This example

clearly supports the coherence theory, rather than the foundational theory.

Most belief revision systems considered in AI, like TMS [Doyle, 1979], ATMS [de Kleer, 1986] and all of their descendants follow FTBR. Also most systems we know of have concentrated mainly on implementation issues. Notable exceptions to this are the work by Reiter and de Kleer [1987], Brown et.al. [1987] and Martins and Shapiro [1988] which analyse the logical foundations of these systems. The only AI system based on CTBR is the Procedural Reasoning System [Georgeff and Lansky, 1986]. The most extensive study of the logical foundations of CTBR has been carried out in the area of philosophical logic by Alchourrón, Gärdenfors and Makinson [1985] (henceforth called the AGM-theory).

The main aim of this paper is to study the semantics of belief revision based on both the coherence and foundational theories. We also briefly discuss situations under which one might prefer coherence theory over foundational theory and vice versa.

The organization of the paper is as follows. Section 2 discusses the semantics of the coherence theory of belief revision. The model theory is based on the possible-worlds framework, and the axiomatization on the AGM-theory. The important result of this section is the soundness and completeness theorem for CTBR. Before one can discuss the semantics of FTBR one needs a precise notion of *justifications* and *justified beliefs*. Section 3 presents the semantics of a *justified belief system* which formalizes the notion of justified beliefs, which interestingly enough, is a second-order concept. The semantics of FTBR is given in section 4 by postulating the *foundational equation*, which is as follows:

$$\text{FTBR} = \text{CTBR} + \text{disbelief propagation.}$$

Section 5 concludes with a brief discussion on the appropriate situations under which one would prefer one theory over the other and the comparison of our work with related work.

The work presented here is a part of the first authors' dissertation. Due to space constraints the proofs of the theorems and the applications of the theory will not be discussed and the interested reader can refer to the complete report [Rao, 1989].

2 Coherence Theory of Belief Revision

The underlying logic on which belief revision is carried out is a modal logic of time-dependent beliefs. The formula $\mathsf{BEL}(t,\phi)$ [1] will be used to denote that the agent believes ϕ at time t. A branching time temporal logic will be used with a standard first-order interpretation for time. Quantifying individual constants into the scope of modal operators is not allowed, but is allowed for time constants. The semantics for such a time-dependent belief system can be given in terms of a slightly modified Kripke interpretation. The satisfaction of belief formulas is given with respect to a world and a time point which acts like an index into the course of events defining that world [Cohen and Levesque, 1987]. The axiomatization is a KD45 axiomatization [Chellas, 1980] together with the axioms of transitivity and backward linearity for time.

Further, we shall treat the BEL operator as a self-belief operator as in Autoepistemic Logic (AEL) [Moore, 1985, Konolige, 1988]. A Kripke interpretation with the above interpretation of beliefs will be called an *autoepistemic Kripke interpretation* (AKI). The attractive feature of such a time-dependent AE belief system is that the stable sets or possible worlds are uniquely determined by the first-order formulas and universal AE beliefs (formulas of the form $\forall\tau\mathsf{BEL}(\tau,\phi)$) of the system. This feature will be exploited in discussing the dynamics of the belief system.

Let the agent at time t and world w receive a first-order formula ϕ from the external world. Ruling out inconsistent possible worlds leaves us with the following three relationships between the formula ϕ and the world w at time t: (1) the agent believes in ϕ at t, (2) the agent believes in $\neg\phi$ at t, and (3) the agent does not believe in ϕ nor $\neg\phi$ at t and hence the agent is *uncommitted* about ϕ. By *dynamics* of belief systems or *belief revision* we mean the process by which an agent moves from one of the three regions (1, 2, or 3) to any of the other two. Ignoring the trivial transitions of remaining in the same state we are left with six different transitions:

1. **Expansion:** Uncommitted state $\rightarrow$ Belief in ϕ.
2. **N-Expansion**: Uncommitted state $\rightarrow$ Belief in $\neg\phi$
3. **Contraction**: Belief in $\phi \rightarrow$ Uncommitted state.
4. **N-Contraction**: Belief in $\neg\phi \rightarrow$ Uncommitted state
5. **Revision**: [2] Belief in $\neg\phi \rightarrow$ Belief in ϕ, and
6. **N-Revision**: Belief in $\phi \rightarrow$ belief in $\neg\phi$.

The terminology and approach is an extension of the AGM-theory. While their approach is non-modal and at the meta-level, we introduce modal operators and carry out the analysis at the object level.

Three *dynamic modal operators*, EXP, CON and REV are introduced to denote expansion, contraction and

[1] All modal operators introduced in this paper are also agent-dependent. For the sake of simplicity we ignore the argument denoting agents. The reader can find the detailed version in the complete report [Rao, 1989].

[2] The word "belief revision" will be used in the more general sense to denote dynamics of belief systems and the word "revision" will be used in the more restricted sense as defined above.

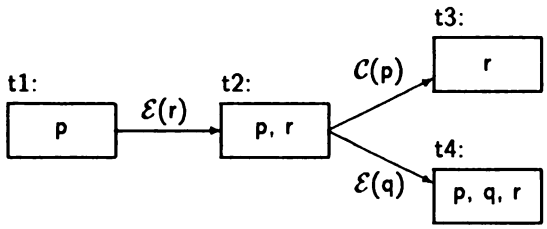


Figure 1: Selecting the closest worlds

revision respectively. Thus EXP(t, ϕ, u) is read as 'the expansion by the agent at t with respect to ϕ is u', where ϕ is a first-order sentence and t and u are time points. In fact, u is the next time point to t. Note that there can be more than one next instant, as the underlying temporal logic is a branching time temporal logic. Strictly speaking there is no need for the modal operator REV as it can be defined by using EXP and CON. All other operations can be defined in terms of EXP and CON.

2.1 Possible-worlds Semantics

The semantics of dynamic operators is based on *selection functions*, which select some possible worlds as being *closer* to the current world than the others [Lewis, 1973]. When the agent performs expansion or contraction, he is said to move into one of these closer worlds and designate these worlds as the worlds of the next time instant(s). Let us say that we have got a proposition p in some world w at time t1 and the agent expands with r. After expansion, she can move into a time point t2, where p and r are true or a time point t3 where only r is true or a time point t4 where p, q and r are true and so on. Amongst the different alternatives, we would like the agent to *choose* only some of them, based on the the principles of minimal change and maximal coherence. According to these principles the selection function for expansion, or *expansion function*, denoted by $\mathcal{E}$, should choose the time point t2, where p and r are true. The other time points are also accessible from t1 but all of them have to pass through t2. Thus t3 can be obtained from t2 after a contraction with respect to p, and t4 from t2 after an expansion with respect to q. The result of the selection function is usually a set of time points. This is illustrated in Figure 1. The rectangular box labelled t1 actually denotes the formulas in world w at time t1. Similarly the box labelled t2 denotes the formulas in world w at time t2 and so on. Whenever it is unambiguous we shall drop the world argument and refer to the formulas by their time index.

The interpretation for this language and the semantics of the dynamic operations are formally defined as follows:

Definition 1 : A *dynamic coherence interpretation* is a tuple, CI = $\langle \mathcal{E}, \mathcal{C}$, AKI$\rangle$, where AKI is an autoepistemic interpretation, $\mathcal{E}$ is an expansion function and $\mathcal{C}$ is a contraction function. The autoepistemic Kripke Interpretation AKI is a tuple, AKI = $\langle$W, $\mathcal{B}$, TU, $\preceq$, U, M$\rangle$, where W is a set of worlds, $\mathcal{B}$ is the accessibility relation for beliefs, TU is a set of time points, $\preceq$ is a partial order on TU, U is a set of individuals and M is the meaning function. The expansion and contraction functions map W, TU and 2^W to 2^{TU}.

The variable assignment VA and term assignment TA are defined as in standard first-order temporal logic.

Definition 2 : A dynamic coherence interpretation CI, satisfies a well-formed formula ψ, at world w and time v with variable assignment VA (written as CI, w, v $\models \psi$[VA]) given the following conditions:

1. (a) CI, w, TA(t) $\models$ EXP(t, ϕ, u)[VA] iff TA(u) $\in$ $\mathcal{E}$(w, TA(t), $||\phi||^{CI}$).
   (b) CI, w, v $\models$ EXP(t, ϕ, u)[VA] iff CI, w, TA(t) $\models$ EXP(t, ϕ, u)[VA].
2. (a) CI, w,TA(t) $\models$ CON(t, ϕ, u)[VA] iff TA(u) $\in$ $\mathcal{C}$(w, TA(t), $||\phi||^{CI}$).
   (b) CI, w, v $\models$ CON(t, ϕ, u)[VA] iff CI, w, TA(t) $\models$ CON(t, ϕ, u)[VA].
3. CI, w, v $\models$ REV(t, ϕ, u)[VA] iff CI, w, v $\models$ CON(t, $\neg\phi$, x)[VA] and CI, w, v $\models$ EXP(x, ϕ, u)[VA].
4. The satisfaction of all the other wff's of this language are defined in the usual way.

The notation $||\phi||^{CI}$ stands for all the worlds of the interpretation CI that satisfy ϕ. More formally, $||\phi||^{CI}$ = {w | CI, w, t $\models \phi$ for all t $\in$ TU}. In the above definition ϕ is a first-order sentence. The unique properties of AEL allows us to restrict our attention to first-order sentences rather than arbitrary belief formulas, while performing expansion, contraction and revision. Conditions 1.a and 2.a give the satisfaction of the dynamic formulas at a given world and the time point in which the operation is being carried out. Conditions 1.b and 2.b provide the satisfaction conditions at all other time points. These conditions are required as we need to evaluate the truth of the dynamic formulas, not only at the time point in which they are being carried out, but also all the other time points as well.

Although we have given the semantics of expansion and contraction in terms of the selection functions, $\mathcal{E}$ and $\mathcal{C}$, we have not imposed enough semantic conditions on these selection functions in order to select the closest worlds. This is done in the next section by providing the axioms, whose analogous semantic conditions constrain the worlds. A constructive procedure for $\mathcal{E}$ and $\mathcal{C}$ is given elsewhere [Rao and Foo, 1988].

2.2 Axiomatization

The axioms of expansion, contraction, and revision are listed in Appendix - I. The **Axiom of Inclusion (AE1)**, **Axiom of Preservation of Beliefs (AE2)**,

Axiom of Trivial Expansion (AE3), and **Axiom of Monotonicity (AE4)** enforce the beliefs that the agent *should* hold after expansion. But they do not rule out the possibility of the agent acquiring beliefs which are in no way related to the belief, say ϕ, with respect to which the expansion is being carried out nor the original world. In other words we need to state that the *only* beliefs after expansion are the beliefs before expansion and the belief in ϕ and all its consequences. This is expressed by the **Axiom of Minimization of Beliefs (AE5)**. We also need an inference rule (RE1) which states that expansions with respect to equivalent formulas are equivalent.

The **Axiom of Exclusion (AC1)**, **Axiom of Minimization of Beliefs (AC2)**, and **Axiom of Trivial Contraction (AC3)** are satisfied by contraction and are similar to the corresponding axioms of expansion. The axiom of monotonicity is not satisfied by contraction. This is because contraction reduces the beliefs of the agent and is therefore non-monotonic in nature. The axiom (AC4) states that whatever is *lost* during contraction is *recovered* during expansion. As we gain as little as possible during expansion (due to the Axiom of Minimization of Beliefs), we have to loose as little as possible for the axiom (AC4) to hold. This implies that the axiom (AC4) performs the function of miniming non-beliefs and is called the **Axiom of Minimization of Non-beliefs (AC4)**. We also need an inference rule (RC1) which is analogous to (RE1).

The **Axiom of Revision(AR1)** which incorporates the Levi identity [Gardenfors, 1988] states that revision with respect to ϕ is equivalent to contraction with respect to $\neg\phi$ followed by expansion with respect to ϕ.

Axioms (AE1) - (AE5) and (AC1) - (AC4) capture the proof-theoretic notion of *minimal change* and *maximal coherence* for expansion and contraction, respectively. Analogous to these axioms one can specify semantic conditions that capture the semantic notion of *closest* possible worlds [Rao, 1989]. The class of models whose $\mathcal{E}$ ($\mathcal{C}$) selection function satisfies these semantic conditions are called $\mathcal{E}$-models ($\mathcal{C}$-models). We shall refer to the modal system KD45 together with the above axioms and inference rules as the *Coherence Modal System* or CS-modal system. We define the class of coherence $\mathcal{D}$-models as models whose $\mathcal{B}$ relation is a $\mathcal{B}$-model (i.e. $\mathcal{B}$ is serial, transitive and euclidean), $\mathcal{E}$ is a $\mathcal{E}$-model and $\mathcal{C}$ is a $\mathcal{C}$-model. The proof of the following theorem and the semantic conditions are given in the complete report [Rao, 1989].

Theorem 1 : The coherence modal system or the CS-modal system is sound and complete with respect to the class of coherence $\mathcal{D}$-models.

The above result is very important as it establishes, for the first time, a sound and complete theory of belief revision.

3 Justified Belief System

The belief system discussed in section 2 is inadequate for FTBR as it does not model *justifications* for beliefs. Beliefs are not only time-dependent, but also justification-dependent. The word *justification* is used in a somewhat broader sense than the usage of justifications in TMS. In fact the usage here is closer to the usage in some philosophical circles [Pappas and Swain, 1978]. A justification for a formula ϕ is a *reason* for believing the formula ϕ. A reason that justifies a formula ϕ will be taken to be any wff. For example, a person John believes that the roads will be icy because he has sufficient reason to believe so - namely that the forecast predicts snow. Thus the formula roads(icy) is justified by the formula forecast-predicts(snow).

It is not necessary that a formula be justified by a single atomic formula. It can be any complex formula. For example β can be justified by the formula $\alpha \wedge (\alpha \rightarrow \beta)$. A formula can also have multiple justifications. For example, in addition to the above justification the formula β could also be justified by $\gamma \wedge (\gamma \rightarrow \beta)$.

The above justification $\alpha \wedge (\alpha \rightarrow \beta)$ deductively implies the formula β, hence it is called a *deductive justification*. TMS and ATMS make use of deductive justifications extensively. But all justifications need not be deductive, some of them can be *inductive justifications*. Consider the belief of John that the train will be on time today. This belief of John may not be justified by John being absolutely sure that the train will be on time or by some other beliefs which deductively entail the train being on time. It might be justified by an inductive argument, namely that John has been catching the train for over an year and it has always come on time, so he believes that it will come on time today also. We shall allow both deductive and inductive justifications of formulas. Although we allow inductive justifications we do not describe how such inductive justifications can be obtained. For obtaining such justifications one needs sophisticated mechanisms of hypothesization and generalization, which is beyond the scope of this paper. [3]

Justifications are also time dependent. Today one might believe that rising commodity prices is sufficient to justify the rising value of the Aussie dollar, but later on one might discover that there are many more factors involved in the rise of the dollar. This will result in a corresponding change in justifications.

By the very definition of foundational theory all the formulas are either incorrigibly justified (i.e. they are self-evident) or are justified by a non-circular and terminating chain of justifications. The onus of providing such non-circular justifications is on the user. Some

[3] Formal theories of hypothesization is an active area of research in machine learning. Delgrande [1987] describes an elegant formal theory of hypothesization.

would prefer the semantics to handle the circularity. However, this restriction is analogous to the case in classical logic, where the onus of specifying a consistent set of formulas is on the user.

Interestingly enough, an agent is not logically omniscient with respect to his justified beliefs, even though he is omniscient with respect to his beliefs. In other words, the agent's justified beliefs are not closed under implication, not closed under valid implication and the agent does not justifiably believe all valid formulas. This resource-bounded justified belief system will be discussed first. This will be followed by axioms of deductive justifications which force logical omniscience. An interesting aspect of these axioms is that they allow the system to derive the (deductive) justifications, rather than being specified by the user. This partially alleviates the problem of having the user specify the justifications for beliefs.

Unfortunately, the need for reasoning about chains of justifications makes the justified belief system a second-order modal logic. One needs to quantify over justifications which are themselves modal formulas. As a result the semantics as well as the axiomatization will be second-order. The main disadvantage of such a logic is that one can give only a weak completeness result. However the logic does shed some new light into the nature of the foundational theory of belief revision.

To summarize, we shall consider justifications to be complex formulas which are time dependent. [4] Also we shall assume that all justifications are either incorrigibly justified or have a non-circular, terminating chain of justifications ending at incorrigibly justified beliefs.

3.1 Syntax of Justified Belief System

The syntax of justified belief system is given by a second-order language. Apart from the standard primitives and operators of the time-dependent belief system (discussed in section 2), the language consists of n-place *predicate variables* and the modal operators JUSTIFY and JBEL. The set of n-place predicate letters, say P, of first-order logic will be referred to as n-place *predicate constants*. The modal operator JUSTIFY(t, ϕ, α) denotes that α is the justification of ϕ at time t. JBEL(t, ϕ, α) denotes that the agent justifiably believes in ϕ at t because of α. The *well-formed formulas (wff)* of the language can now be defined as follows:

1. If $s_1,\ldots,s_n$ are first-order terms and ϕ is an n-place predicate variable or an n-place predicate constant then $\phi(s_1,\ldots,s_n)$ is a wff.
2. If t and u are temporal terms, τ is a temporal variable, ξ is a predicate variable, ϕ_1, ϕ_2 are wff, $\neg\phi_1$, $\phi_1 \wedge \phi_2$, BEL(t, ϕ_1), JUSTIFY(t, ϕ_1, ϕ_2), JBEL(t, ϕ_1, ϕ_2), $\forall\xi(\phi_1)$ $\forall\tau(\phi_1)$ are wff.

Some of the wff of the language are as follows: (a) JBEL(t, ϕ, true) - belief in ϕ at t is incorrigibly justified, (b) JBEL(t, ϕ, $\neg$BEL(t, α)) and (c) $\forall\xi_1\ \exists\xi_2$ JUSTIFY(t, ξ_1, ξ_2).

3.2 Semantics of Justified Belief System

The semantics of justified belief system is an extension of the Kripke interpretation. To provide the semantics of the modal operators JUSTIFY and JBEL, we introduce a *justification function*, denoted by $\mathcal{J}$. As both the modal operators are time dependent, the justification function $\mathcal{J}$ is also time dependent. As formulas are evaluated at each world, the function $\mathcal{J}$ depends on the world at which it is being evaluated. The main purpose of the justification function is to map a given formula to a set of formulas, each one of which is a justification for the given formula. It is convenient to express a formula as a set of worlds where the formula is true. Thus justification functions are mappings from a set of worlds to a set of set of worlds. More formally, the justification function $\mathcal{J}$, is a mapping from W (worlds), TU (time points) and 2^W (set of worlds) to 2^{2^W} (set of set of worlds).

In a Kripke interpretation one specifies the truth value of atomic formulas for each world, at each time point, and the accessibility relation $\mathcal{B}$ for beliefs. Based on this the evaluation of all other formulas are carried out. In a justified belief system, in addition to the above, one has to specify the $\mathcal{J}$ function, giving the justification for each formula (not just atomic formulas). [5] This requirement is contrary to the traditional philosophical thinking that the meaning of complex sentences depend only on the truth conditions of atomic formulas. We treat justification functions at the same primitive level as giving the truth condition to atomic formulas. The justifications are part of the meaning of sentences of the form 'the agent justifiably believes ϕ because of α' and 'the agent is justified in ϕ by α' and hence should be stated before hand. Our position is more in line with the philosophy of Pollack [1974] who argues that only certain concepts can be analyzed in terms of truth conditions alone and all other concepts need justification conditions as well as truth conditions. Such concepts are called *ostensive concepts*. According to his theory of epistemic justification, one has to specify the justification conditions as well as the truth conditions for analyzing the meaning of sentences.

The process of evaluating the formulas of the justified belief system is illustrated with an example.

[4] Justifications are also agent-dependent. Once again we ignore this dependency for the sake of simplicity.

[5] One can introduce a language which is less expressive than the one given here, by preventing nesting of justifications. For such a language it is sufficient to specify the justifications of only the atomic formulas.

Example 1 : Consider the situation where the formulas platypus(platty) and platypus(platty) → lives(platty, oz) are incorrigibly justified and the formula lives(platty, oz) is justified by the formula platypus(platty) ∧ (platypus(platty) → lives(platty, oz) for the agent at time t1 and world W0. From this it is not very difficult to conclude that the formulas JUSTIFY(t1, platypus(platty), true), JUSTIFY(t1, platypus(platty) → lives(platty, oz), true) and JUSTIFY(t1, lives(platty, oz), platypus(platty) ∧ (platypus(platty) → lives(platty, oz))) will be satisfiable. Also one would expect the formula platypus(platty) ∧ (platypus(platty) → lives(platty, oz)) to be justified by the conjunction of the formulas that justify platypus(platty) and platypus(platty) → lives(platty, oz). In other words, from the above one would expect JUSTIFY(t1, platypus(platty) ∧ (platypus(platty) → lives(platty, oz)), true) to be satisfiable.

In order to provide the meaning of BEL and JBEL formulas, the accessibility relation $\mathcal{B}$ and the truth value of the atomic formulas at all the accessible worlds have to be provided. Let us assume that platypus(platty) and platypus(platty) → lives(platty, oz) are true in the current world W0 and all the accessible worlds. Therefore, lives(platty, oz) is true in the current world and all the accessible worlds. Thus the formulas BEL(t1, platypus(platty)), BEL(t1, platypus(platty) → lives(platty, oz)) and BEL(t1, lives(platty, oz)) are satisfiable. Now how does one evaluate the claim that belief in lives(platty, oz) is justified? In other words, how does one evaluate the formula JBEL(t1, lives(platty, oz), platypus(platty) ∧ (platypus(platty) → lives(platty, oz)))? It is reasonable to stipulate that the above formula is satisfiable iff the following conditions are satisfied,

1. lives(platty, oz) is believed by the agent at time t1,
2. lives(platty, oz) is justified by platypus(platty) ∧ (platypus(platty) → lives(platty, oz)) for the agent at time t1, and
3. the justification platypus(platty) ∧ (platypus(platty) → lives(platty, oz)) is justifiably believed by the agent at time t1,

Notice that the evaluation of JBEL formulas is recursive. The recursion ends with incorrigibly justified beliefs. If we impose the condition that JBEL(t1, true, true) is always true, then the justified belief in condition 3 is satisfied. As condition 1 and 2 are also satisfied the formula JBEL(t1, lives(platty, oz), platypus(platty) ∧ (platypus(platty) → lives(platty, oz))) is also satisfiable. □

In the above example the formula lives(platty, oz) had to be justified only in the current world (condition 2). It was not required that lives(platty, oz) be justified by platypus(platty) ∧ (platypus(platty) → lives(platty, oz))) in all the $\mathcal{B}$-accessible worlds. Justifications which satisfy the latter condition will be called *strong justifications* as opposed to *weak justifications* of the example.

The interpretation JI of the justified belief system is defined as follows:

Definition 3 : The second order interpretation JI, is an infinite sequence JI = ⟨W, $\mathcal{B}$, TU, $\preceq$, U, M, U_1, U_2, ...⟩ where

1. W, $\mathcal{B}$, TU, $\preceq$ and U are as defined before.
2. Each U_N (n = 1,2,...) is a nonempty set of n+2-tuple ⟨w, t, s_1, ..., s_n⟩, where t ∈ TU, w ∈ W and each of s_1, ..., s_n ∈ U.
3. M is a tuple M = ⟨M1, M2, M3⟩ such that the following conditions are satisfied:
   (a) For any individual constant c, M1(c) ∈ U
   (b) For any n-place predicate constant p of P, M2(p) ∈ U_N (n = 1, 2, ..)
   (c) For any itemporal constant t, M3(t) ∈ TU

The variable assignment VA is given for predicate variables also. Thus VA = ⟨VAT, VAV, VAP⟩, where VAT, VAV and VAP are the variable assignments for temporal variables, individual variables and predicate variables, respectively. For example, the variable assignment for predicate variables is defined as follows: for any n-place predicate variable ξ, VAP(ξ) ∈ U_N. The term assignment for temporal and first-order terms are defined in the usual way. For predicate constants and variables they are an extension of the first-order case.

Definition 4 : A justified interpretation JI, satisfies a wff ψ under the variable assignment VA at world w and time v (written as JI, w, v $\models \psi$[VA]) given the following conditions,

1. JI, w, v $\models \phi(s_1,...,s_n)$ iff ⟨w, v, TA(s_1), ..., TA(s_n)⟩ ∈ TA(ϕ), where ϕ is a predicate constant or predicate variable.
2. (a) JI, w, TA(t) $\models$ JUSTIFY(t, ϕ, α)[VA] iff $\|\alpha\|_t^{JI} \in \mathcal{J}(w, TA(t), \|\phi\|_t^{JI})$.
   (b) JI, w, v $\models$ JUSTIFY(t, ϕ, α)[VA] iff JI, w, TA(t) $\models$ JUSTIFY(t, ϕ, α)[VA].
3. (a) JI, w, TA(t) $\models$ JBEL(t, ϕ, α)[VA] iff
      i. JI, w, TA(t) $\models$ BEL(t, ϕ)[VA],
      ii. JI, w, TA(t) $\models$ JUSTIFY(t, ϕ, α)[VA] and
      iii. JI, w, TA(t) $\models \exists\beta$JBEL(t, α, β)[VA].
   (b) JI, w, v $\models$ JBEL(t, ϕ, α)[VA] iff JI, w, TA(t) $\models$ JBEL(t, ϕ, α)[VA].
4. The satisfaction of formulas of the form $\neg\phi$, $\phi_1 \wedge \phi_2$, $\forall x(\phi(x))$ and BEL(t, ϕ) are as defined earlier for the Kripke interpretation KI.

The most interesting aspect of the above semantics is the recursive nature of the operator JBEL. Condition iii of the definition recursively goes through the chain of justifications, checking if each one of the justifications is justifiably believed. According to the foundational theory such a chain of justification should finally terminate at an incorrigibly justified belief. Condition

3.a.i of an incorrigibly justified belief will be JBEL(t, true, true), which we take to be trivially satisfied. [6]

Condition 3 of the above definition captures the notion of weak justification. Strong justification is captured by strengthening condition 3.a.ii of the above definition. This modification is given below.

3'. (a) JI, w, TA(t) $\models$ JBEL(t, ϕ, α)[VA] iff

i. JI, w, TA(t) $\models$ BEL(t, ϕ)[VA],

ii. JI, w, TA(t) $\models$ BEL(t, JUSTIFY(t, ϕ, α))[VA] and

iii. JI, w, TA(t) $\models \exists\beta$JBEL(t, α, β)[VA].

Condition 3'.a.ii ensures that ϕ is justified by α not only in the current world, but also in all the $\mathcal{B}$-accessible worlds.

3.3 Axiomatization

The axioms of second-order logic are similar to that of first-order logic, except that wherever individual variables and individual constants occur, predicate variables and predicate constants can also occur. In addition the following *comprehension axiom schema* [Robbin, 1969] is needed. This axiom states that every wff defines a predicate.

> $\exists\xi\forall x_1,..,x_n\ \xi(x_1, ..., x_n) \equiv \phi$, where ξ is any n-place predicate variable not occurring free in ϕ and $x_1, ..., x_n$ are individual variables.

We shall refer to the above set of modified second-order axioms as (AJ1) and the modified second-order inference rules as (RJ1).

The rest of the axioms of justified belief system is given in Appendix-II. Axiom (AJ2) captures the semantics of justified beliefs. It states that the agent at time t justifiably believes ϕ because of α iff the agent believes in ϕ, ϕ is justified by α, and α is justifiably believed because of β. There is no need for any semantic condition for (AJ2) as it directly captures the semantic conditions 3.a and 3.b of JBEL.

Axiom (AJ2) captures the notion of weak justification. The semantic conditions 3'.a and 3.b of strong justifications is captured by axiom (AJ2'). According to the foundational theory all justified beliefs will terminate in an incorrigibly justified belief. By axiom (AJ2) all incorrigibly justified beliefs end with a formula of the form JBEL(t, true, β). We shall assume that the formula true is trivially justified by true itself. This is given by axiom (AJ3). In addition to the above axioms, one more inference rule is required. As we are following the minimal model semantics [Chellas, 1980] for the $\mathcal{J}$-function, we require the inference rule (RJ2) which states that if a formula is justified by another formula α then it is also justified by a semantically equivalent form of α.

We shall refer to the axioms (AJ1)-(AJ3) and inference rules (RJ1) and (RJ2) as the *JS-modal system* of a weakly justified belief system. Models of this belief system whose $\mathcal{J}$ function satisfies certain conditions [Rao, 1989] and whose $\mathcal{B}$-relation is a $\mathcal{B}$-model, is called a $\mathcal{J}$-model. Replacing definition 3 by 3' gives the *JS'-modal system* of a *strongly justified belief system* and a corresponding $\mathcal{J}'$-model.

If in a second-order interpretation JI, each U_N (n=1,2,...) consists of *all* the n-place predicates on U, and the interpretation assigns the identity relation among the individuals of U to the predicate constant '=', then it is called a *principal interpretation* [Rogers, 1971]. The corresponding model is called a *principal model*. They are the standard interpretations/models of second-order logic. The other interpretations/models are those in which either not all of the domains of n-place predicates are complete in the above sense, or some predicate other than the identity relation is assigned the symbol '='. Such interpretations are called *secondary interpretations* [Rogers, 1971] and their corresponding models, *secondary models*. Unfortunately the completeness of second-order logic can be shown only with respect to both the principal and secondary models, and cannot be shown with respect to the principal models alone. As secondary interpretations are nonstandard interpretations, the resulting completeness result is considered a *weak completeness* result. The soundness and completeness of the justified belief system can now be stated as follows:

Theorem 2 : The JS-modal system (JS'-modal system) is sound and complete with respect to the class of all principal and secondary $\mathcal{J}$-models ($\mathcal{J}'$-models).

3.4 Deductive Justifications

It is interesting to note that the justified belief systems introduced in the previous sections are not logically omniscient. The beliefs of the agent are still logically omniscient, but the justified beliefs of the agent are not logically omniscient. This is because justified beliefs are based on minimal model semantics, with very little restrictions on how the justifications should be constructed. Thus it is possible for the following formulas to be satisfiable.

1. JBEL(t, p, r) $\wedge$ JBEL(t, p $\rightarrow$ q, s) $\wedge \neg$JBEL(t, q, m) is satisfiable if m is the only formula that justifies q and m is not believed by the agent at t.
2. p $\rightarrow$ (q $\rightarrow$ p) is valid but JBEL(t, p, r) $\wedge \neg$JBEL(t, q $\rightarrow$ p, s) is satisfiable if s is the only formula that satisfies q $\rightarrow$ p and s is not believed by the agent at time t.
3. p $\vee \neg$p is valid but JBEL(t, p $\vee \neg$p, r) is satisfiable if r is the only formula that justifies p $\vee \neg$p and r is not believed by the agent at time r.

Various conditions can be imposed on the justifications to reflect how these formulas have been derived. As the conditions reflect the deductive rules of logic they will be called *deductive justification axioms*. The

[6] An appropriate axiom will be introduced in the next section to make this so.

axioms discussed in the previous section do not impose any conditions on how the justifications of conjunctive, disjunctive and negated formulas relate to the justifications of atomic formulas. These conditions can be imposed by adopting axioms (AJ4)-(AJ6). Axiom (AJ4) states that if α justifies ϕ then it cannot justify the negation of ϕ, as well. This prevents logically inconsistent states from being justified. Axiom (AJ5) states that the conjunction of formulas ϕ_1 and ϕ_2 is justified by the conjunction of their respective justifications α_1 and α_2. Axiom (AJ6) states the same for disjunction of formulas.

It is interesting to note, that in axiom (AJ5), if ϕ_2 is the negation of ϕ_1, then $\alpha_1 \wedge \alpha_2$ gives the reason for the inconsistent formula $\phi_1 \wedge \neg\phi_1$. One can trace the justifications of $\alpha_1 \wedge \alpha_2$, the justifications for its justifications and so on, till one ends up at incorrigible justifications which are responsible for the inconsistency. This is similar to the way TMS recursively goes through the chain of justifications to identify the culprits causing the inconsistency. This process is called dependency-directed backtracking in TMS.

Axiom (AJ7) states how deductive justifications are to be created. If ϕ_1 is justified and $\phi_1 \rightarrow \phi_2$ is justified, then ϕ_2 is justified by the formula $\phi_1 \wedge (\phi_1 \rightarrow \phi_2)$. This is nothing but the principle of modus ponens. The next two justifications are the axioms of introspection of justifications. Axiom (AJ8) states that a formula ϕ is justified by α iff the fact that it is justified is itself incorrigibly justified. Similarly, axiom (AJ9) states that a formula ϕ is not justified by α iff the fact that it is not justified is itself incorrigibly justified.

In addition to the above axioms, the inference rule (RJ3) which states that all valid formulas are always incorrigibly justified is also needed. This makes the system logically omniscient with respect to valid formulas.

We shall refer to the axioms (AJ1)-(AJ9) and inference rules (RJ1)-(RJ3) as the *deductive JS-modal system* of a weakly justified belief system. Models of this belief system whose $\mathcal{J}$ function satisfies the corresponding semantic conditions and whose $\mathcal{B}$-relation is a $\mathcal{B}$-model, is called a *deductive $\mathcal{J}$-model.* Similarly we have *deductive JS'-modal system* and *deductive $\mathcal{J}'$-models* for a strongly justified belief system. Once again the above modal systems are sound and complete only with respect to both the principal and secondary models.

Theorem 3 : The deductive JS-modal system (deductive JS'-modal system) is sound and complete with respect to the class of all principal and secondary deductive $\mathcal{J}$-models ($\mathcal{J}'$-models).

4 Foundational Theory of Belief Revision

Before discussing the dynamics of justified belief systems, we make a simplifying assumption that justifications *do not change* from one time to another. In other words the addition and removal of justifications will not be discussed. The expansion and contraction functions discussed in section 2 will not be extended to handle justification formulas. They will still be restricted to first-order formulas. Note that this assumption does not prevent the justified beliefs from being dynamic. As justified beliefs depend on justifications and beliefs, we study the dynamics of justified beliefs by keeping the justifications constant, but changing the beliefs and studying the effects of these changes on the justified belief system.

In section 2, the dynamics of belief systems was studied with respect to an autoepistemic belief system, where the belief operator was treated as a self-belief operator. Accordingly, the interpretation of beliefs was with respect to an autoepistemic Kripke interpretation, AKI. Similarly a justified belief system whose belief operator is treated as a self-belief operator will be called a *justified autoepistemic belief system* and the corresponding interpretation will be denoted by AJI. As the dynamics will be discussed with respect to an justified autoepistemic belief system the dynamic operations will be restricted to first-order formulas.

The language under discussion for FTBR is the language of justified belief system together with the dynamic operators of coherence theory, namely EXP, CON and REV and analogous dynamic operators of foundational theory, denoted by FEXP, FCON and FREV. The wff's of the language are defined in the usual way.

4.1 Semantics

The main difference between coherence and foundational theories is that the coherence theory does not concern itself with justifications, while the foundational theory does. As a result, CTBR allows beliefs which are not justified, while FTBR insists that *all beliefs* be *justified beliefs.* In other words, assuming that one starts at time t with all beliefs being justified beliefs, FTBR insists that after expansion/contraction, say at time u, the following expression, which we shall call the *foundational thesis*, is satisfied:

$$\mathsf{BEL}(u, \phi) \rightarrow \mathsf{JBEL}(u, \phi, \alpha).$$

In a FTBR, for any time point t whenever a formula ϕ is believed it is justifiably believed and vice versa.

In order to enforce the foundational thesis, after carrying out expansion/contraction according to the coherence theory, one should check if there are any formulas which are believed but are not justifiably believed. For foundational expansion/contraction all such formulas should be disbelieved, or removed from

the current state of beliefs. This process of removing beliefs which are not justifiably believed should be carried out till all the beliefs in the system are justified beliefs. This process is called *disbelief propagation* [Martins and Shapiro, 1988]. In other words, one can postulate the following equation, which links foundational and coherence theories:

FTBR = CTBR + Disbelief Propagation.

This equivalence provides the semantics and axiomatization of FTBR. This approach has the advantage of directly comparing both theories of belief revision. However, an independent characterization of FTBR will help in verifying the foundational equation.

As in section 2 the semantics of the operators, FEXP, FCON, and FREV are given with respect to two selection functions $\mathcal{FE}$ and $\mathcal{FC}$, called the *foundational expansion function* and *foundational contraction function* respectively. Both these functions operate on the current time point, world and a set of worlds where a given formula is true, and map them to a set of time points. In section 2, we saw that the selection functions $\mathcal{E}$ and $\mathcal{C}$, selected certain time points to be more *closer* than others based on the two main principles of minimal change and maximal coherence. The selection functions $\mathcal{FE}$ and $\mathcal{FC}$ also select certain time points to be more *closer* than others. But the notion of *closeness* in the foundational theory depends on the following three criteria:

1. minimal change - as few beliefs/non-beliefs as is required should be added or removed
2. maximal coherence - as many beliefs/non-beliefs as possible should be preserved
3. foundational thesis - all beliefs should be justifiably believed

It is quite obvious that what is close according to the coherence theory and selection functions $\mathcal{E}$ and $\mathcal{C}$ may not be close according to the foundational theory and selection functions $\mathcal{FE}$ and $\mathcal{FC}$, and vice versa. The following example illustrates the operation of the selection functions.

Example 2 : Consider the situation given in Figure 2. Let us say that at time t1 one wants to perform a foundational expansion with respect to the formula p. Some of the possible worlds at time points t2, t3, t4, t5 and t6 are shown in the figure. The question is which one of these worlds should the agent choose? The world at time t2 satisfies the criteria 1 and 2 of closeness but not 3. This is because the formulas q and r are believed but not justifiably believed. Worlds at t3 and t4 also do not satisfy condition 3, as they each contain one belief which is not a justified belief, namely q and r respectively. The world at t5 satisfies all the three conditions, and hence should be chosen by the function $\mathcal{FE}$, as the closest world to t1. The world at t6 satisfies the conditions of minimal change

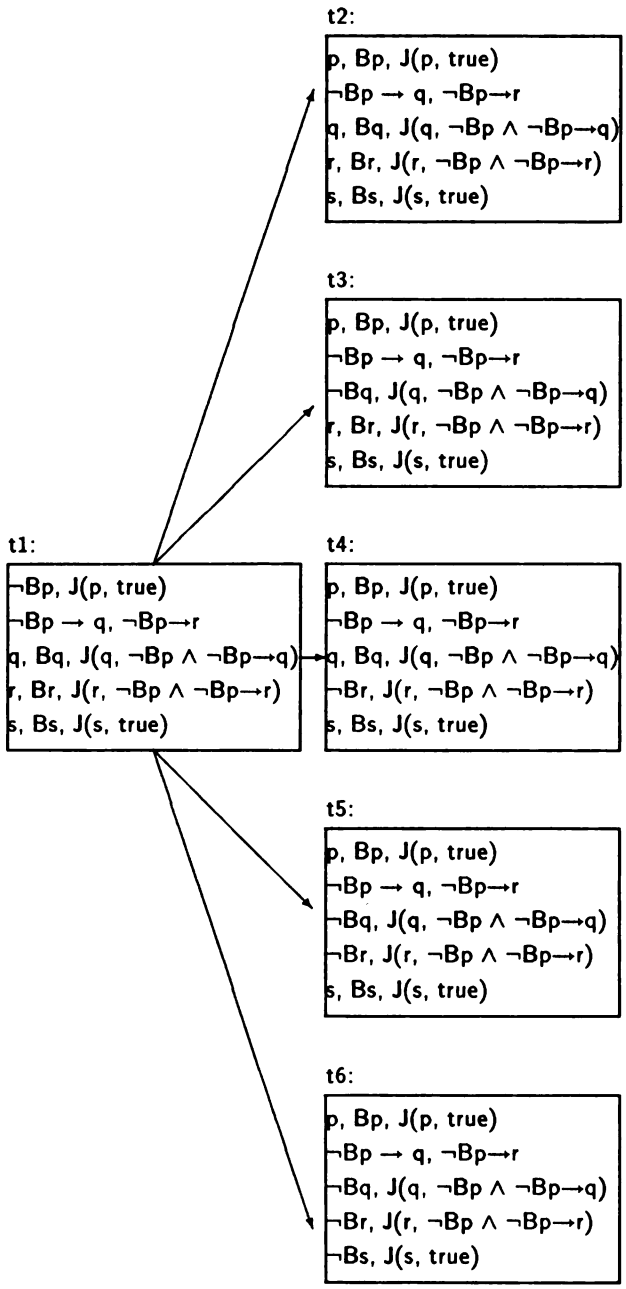


Figure 2: Example of Foundational Expansion

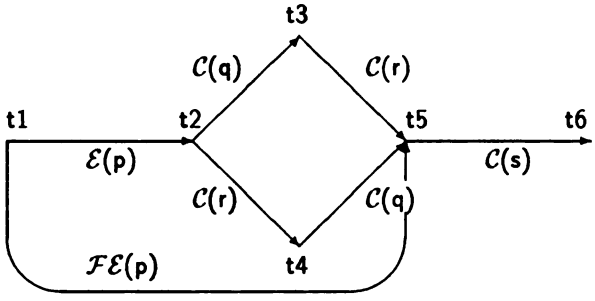


Figure 3: Foundational Equation

and foundational thesis but not maximal coherence, because there is no necessity to give up belief in s. It is clear from the formulas in t1 to t6, that t2 can be obtained from t1 after an expansion with respect to p, t3 from t2 after a contraction wrt q, t4 from t2 after a contraction wrt r, t5 from t3 after a contraction wrt r, t5 from t4 after a contraction wrt q, and t6 from t5 after a contraction wrt s. This is shown in Figure 3. It is not difficult to deduce from the above that foundational expansion involves expansion followed by repeated contractions of formulas which are believed but are not justifiably believed. Hence t5 is obtained from t1 after a foundational expansion wrt p. In both figures the time argument has been ignored in the modal formulas and the selection functions for the sake of clarity. □

In the above example, foundational expansion resulted in an expansion followed by a series of contractions. Similarly foundational contraction results in a series of contractions.

The above example provides an intuitive picture as to how the selection function chooses time points whose worlds satisfy the three conditions of closeness for a foundational theory. Now we formalize this notion. The semantics of the language is given by a *dynamic foundational interpretation*, denoted by FI. It is defined as follows:

Definition 5 : The *dynamic foundational interpretation*, is a tuple FI = $\langle \mathcal{FE}, \mathcal{FC}, \mathcal{E}, \mathcal{C}, \text{AJI} \rangle$, where

- $\mathcal{FE}$ and $\mathcal{FC}$ map, W (worlds), TU(time points) and 2^W (set of worlds where a formula is true) to 2^{TU} (set of time points).
- $\mathcal{E}$ and $\mathcal{C}$ are the same as in the dynamic coherence interpretation CI, and AJI is the autoepistemic justified interpretation.

The term assignment and variable assignment remain the same. The satisfaction of dynamic formulas of foundational theory, namely FEXP, FCON and FREV are similar to the satisfaction of EXP, CON and REV; the only difference is that the evaluation is with respect to the interpretation FI, rather than CI. The semantics of other formulas remain the same. The different properties of the dynamic operations are captured by specifying certain semantic conditions on the selection functions and adopting certain axioms. This is done in the next section.

4.2 Axiomatization

In this section we shall provide an axiomatization for foundational expansion and contraction, which will in turn specify the semantic conditions that are to be imposed on the selection functions $\mathcal{FE}$ and $\mathcal{FC}$. The axioms are listed in Appendix-III.

As we saw earlier foundational expansion is equivalent to expansion followed by repeated removal of beliefs which are not justifiably believed, until there are no such beliefs. Also foundational contraction is equivalent to contraction followed by repeated removal of beliefs which are not justifiably believed. Thus FEXP is defined in terms of FCON, and FCON is recursively defined in terms of itself. These axioms are expressed by **Axiom of Disbelief Propagation in Foundational Expansion (AFE1)** and the corresponding axiom for foundational contraction (AFC1). The first disjunct of both the axioms act like boundary conditions and are true when all beliefs are justified beliefs. The second disjunct performs repeated contractions of unjustified beliefs. **Axiom of Foundational Revision (AFR1)** is similar to (AR1). In addition the inference rules (RFE1) and (RFC1) are also needed. The axioms are given in Appendix - III and semantic conditions in the complete report [Rao, 1989]. The class of models whose $\mathcal{FE}$ ($\mathcal{FC}$) satisfies these semantic conditions will be called the *$\mathcal{FE}$-model* (*$\mathcal{FC}$-model*).

The CS-modal system and the deductive JS-modal system together with the axioms and inference rules of foundational expansion, contraction and revision will be called the *foundational modal system* or FS-modal system. We define the class of *foundational $\mathcal{D}$-model* as models which are coherence $\mathcal{D}$-models and dedcutive $\mathcal{J}$-models, and whose selection function $\mathcal{FE}$ is a $\mathcal{FE}$-model and $\mathcal{FC}$ is a $\mathcal{FC}$-model. The soundness and weak completeness of this system is given below:

Theorem 4 : The FS-modal system is sound and complete with respect to the class of all principal and secondary, foundational $\mathcal{D}$-models.

5 Comparison and Conclusion

The philosophical debate on whether the coherence theory models the behavior of a rational agent better than the foundational theory or vice versa, is inconclusive. However this need not deter AI researchers from adopting either one of them. From an AI persepective, it seems as if CTBR is better suited for dynamic, real-time domains, while FTBR can handle the more traditional domains better than CTBR. There are two main reasons for prefering CTBR over FTBR for real-time domains:

- The process of disbelief propagation is time consuming. In real-time domains where a guaranteed response time is required (as in PRS [Georgeff and Lansky, 1986]) it is preferable to postpone disbelief propagation till it is explicitly requested by the agent, as is done in CTBR.
- In planning systems which interleave plan formulation and plan execution, it is undesirable (and in some cases impossible) to disbelieve the currently unjustified formulas, as was illustrated in the RCS example in the introduction.

The major difference between the CTBR of Alchourrón *et al.* [1985] and the theory described in section 2 are as follows: (a) the object-level logic of AGM theory was a simple propositional logic, whereas in our case it is a AE belief system. (b) the dynamics was captured at a meta-level in the AGM theory. This prevented them from having strong soundness and completeness results as outlined here. (c) AGM theory was restricted only to the coherence theory, whereas we have shown that the principles can be carried over to the foundational theory as well.

Martins and Shapiro [1988] divide the belief revision problem into five different parts - (a) the inference problem, (b) the nonmonotonicity problem, (c) dependency recording, (d) disbelief propagation and (e) revision of beliefs. The inference problem is concerned with deriving new beliefs, given a set of premises and certain deductive constraints. The Clause Maintenance System of Reiter and de Kleer [1987] and the algebraic method of Brown *et al.* [1987] formalize this problem. In our case this problem can be stated as follows: how to derive justified beliefs given a set of incorrigibly justified beliefs and certain deductive constraints. The deductive justification axioms provide the constraints and the satisfaction of justified beliefs is given by the axiom (AJ2). The nonmonotonicity problem is solved by allowing formulas to be justified by both beliefs and lack of beliefs in other formulas. Like TMS the dependecy recording in our case is justification-based. The process of disbelief propagation is captured by the foundational axioms of expansion and contraction. In both TMS and ATMS the revision of beliefs is carried out after an explicit declaration of a contradiction. In our case the process of revision removes the causes for inconsistency, if any, by contraction and then adds or expands with the new belief. This avoids reasoning in the presence of contradiction. The work of Reiter and de Kleer, and Brown *et al.* fail to provide a logical analysis of all the above problems discussed by Martins and Shapiro. The SWM system of Martins and Shapiro solves all the problems, except (b), by providing a proof theory based on relevance logic. The FTBR outlined here provides a model theory and a proof theory, and demonstrates the soundness and weak completeness of such a theory. The systems TMS, ATMS, CMS and SWM all perform disbelief propagation and are therefore based on the foundational theory.

The CTBR outlined here could be implemented as a theorem proving system. However a more fruitful approach would be to modify existing TMS-like systems. TMS essentially carries out dependency-directed backtracking to identify the culprits causing a contradiction, removes one of them and then performs disbelief propagation. A TMS based on coherence theory would perform dependency-directed backtracking to identify culprits causing $\neg\phi$, contract them and then add ϕ in order to perform revision with respect to ϕ. However what remains to be shown is a formal proof that the above method faithfully reflects the model theory and proof theory outlined in section 2. A similar proof is also needed for the FTBR.

Acknowledgement: We would like to thank Mike Georgeff and the reviewers of this paper for their helpful comments. The first author was supported by the Sydney University Postgraduate Scholarship and the Research Foundation for Information Technology. Thanks are also due to IBM Corp. for their generous support during our stay at IBM T.J. Watson Research Center and IBM Thornwood respectively.

References

[Alchourrón *et al.*, 1985] C. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction functions and their associated revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[Brown Jr. *et al.*, 1987] A. L. Brown Jr., D. E. Gaucas, and D. Benanav. An algebraic foundation for truth maintenance. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, volume 2, pages 973–980, Milan, Italy, 1987.

[Chellas, 1980] B. F. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, 1980.

[Cohen and Levesque, 1987] P. R. Cohen and H. J. Levesque. Persistence, intention, and commitment. Technical Report 415, Artificial Intelligence Center, SRI International, Menlo Park, California, 1987.

[de Kleer, 1986] J. de Kleer. An assumption-based tms. *Artificial Intelligence*, 28:127–162, 1986.

[Delgrande, 1987] J. P. Delgrande. An approach to default reasoning based on first-order conditional logic. In *Proceedings of the Sixth National Conference on Artificial Intelligence(AAAI-87)*, pages 340–345, 1987.

[Doyle, 1979] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States.* Bradford Book, MIT Press, Cambridge, Mass., 1988.

[Georgeff and Lansky, 1986] M. P. Georgeff and A. L. Lansky. A system for reasoning in dynamic domains: Fault diagnosis on the space shuttle. Technical Report 375, Artificial Intelligence Center, SRI International, Menlo Park, California, 1986.

[Harman, 1986] G. Harman. *Change in View: Principles of Reasoning.* A Bradford Book, The MIT Press, Cambridge, Mass., 1986.

[Konolige, 1988] K. Konolige. On the relation between default and autoepistemic logic. *Artificial Intelligence*, 35(3):343–382, 1988.

[Lewis, 1973] D. Lewis. *Counterfactuals.* Harvard University Press, Cambridge, Mass., 1973.

[Martins and Shapiro, 1988] J. P. Martins and S. C. Shapiro. A model for belief revision. *Artificial Intelligence*, 35:25–79, 1988.

[Moore, 1985] R. C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25, 1985.

[Pappas and Swain, 1978] G. S. Pappas and M. Swain, editors. *Essays on Knowledge and Justification.* Cornell University Press, Ithaca, New York, 1978.

[Pollack, 1974] J. L. Pollack. *Knowledge and Justification.* Princeton University press, Princeton, New Jersey, 1974.

[Rao and Foo, 1988] A. S. Rao and N. Y. Foo. Dynamics of an autoepistemic belief system. In *Proceedings of the IEEE Conference on AI Theory and Applications*, Hong Kong, 1988.

[Rao, 1989] A. S. Rao. Dynamics of belief systems: A philosophical, logical and ai perspective. Technical Report 2, Australian Artificial Intelligence Institute, Melbourne, Australia, 1989.

[Reiter and de Kleer, 1987] R. Reiter and J. de Kleer. Formal foundations of assumption-based truth maintenance systems: Preliminary report. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 183–188, Seattle, Washington, 1987.

[Robbin, 1969] J. W. Robbin. *Mathematical Logic: A First Course.* W. A. Benjamin, Inc., New York, NY, 1969.

[Rogers, 1971] R. Rogers. *Mathematical Logic and Formalized Theories.* North-Holland Publishing Company, Amsterdam, 1971.

Appendix - I: Coherence Theory of Belief Revision

Axioms for Expansion

(AE1) $EXP(t,\phi,u) \rightarrow BEL(u,\phi)$

(AE2) $EXP(t,\phi,u) \rightarrow (BEL(t,\alpha) \rightarrow BEL(u,\alpha))$

(AE3) $BEL(t,\phi) \wedge EXP(t,\phi,u) \rightarrow (BEL(t,\alpha) \equiv BEL(u,\alpha))$.

(AE4) $(BEL(t,\alpha) \rightarrow BEL(v,\alpha)) \wedge EXP(t,\phi,u) \wedge EXP(v,\phi,y) \rightarrow (BEL(u,\beta) \rightarrow BEL(y,\beta))$.

(AE5) $BEL(u,\phi) \rightarrow BEL(t,\phi) \vee (EXP(t,\alpha,u) \wedge BEL(u, BEL(u,\alpha) \rightarrow \phi))$.

(RE1) From $\vdash \phi_1 \equiv \phi_2$ infer $\vdash EXP(t,\phi_1,u) \equiv EXP(t,\phi_2,u)$.

Axioms for Contraction

(AC1) $CON(t,\phi,u) \rightarrow \neg BEL(u,\phi)$

(AC2) $BEL(u,\phi) \rightarrow BEL(t,\phi) \vee (CON(t,\alpha,u) \wedge BEL(u, \neg BEL(u,\alpha) \rightarrow \phi))$.

(AC3) $\neg BEL(t,\phi) \wedge CON(t,\phi,u) \rightarrow (\neg BEL(t,\alpha) \equiv \neg BEL(u,\alpha))$.

(AC4) $CON(t,\phi,u) \wedge EXP(u,\phi,v) \rightarrow (BEL(t,\alpha) \rightarrow BEL(v,\alpha))$.

(RC1) From $\vdash \phi_1 \equiv \phi_2$ infer $\vdash CON(t,\phi_1,u) \equiv CON(t,\phi_2,u)$.

Axiom of Revision

(AR1) $REV(t,\phi,u) \rightarrow CON(t,\neg\phi,v) \wedge EXP(v,\phi,u)$

Appendix - II: Deductive Justified Belief System

(AJ2) $JBEL(t, \phi, \alpha) \equiv BEL(t, \phi) \wedge JUSTIFY(t, \phi, \alpha) \wedge \exists\beta JBEL(t, \alpha, \beta)$.

(AJ2') $JBEL(t, \phi, \alpha) \equiv BEL(t, \phi) \wedge BEL(t, JUSTIFY(t, \phi, \alpha)) \wedge \exists\beta JBEL(t, \alpha, \beta)$.

(AJ3) $JBEL(t, true, true)$.

(AJ4) $JUSTIFY(t, \phi, \alpha) \rightarrow \neg JUSTIFY(t, \neg\phi, \alpha)$.

(AJ5) $JUSTIFY(t, \phi_1, \alpha_1) \wedge JUSTIFY(t, \phi_2, \alpha_2) \rightarrow JUSTIFY(t, \phi_1 \wedge \phi_2, \alpha_1 \wedge \alpha_2)$.

(AJ6) $JUSTIFY(t, \phi_1, \alpha_1) \wedge JUSTIFY(t, \phi_2, \alpha_2) \rightarrow JUSTIFY(t, \phi_1 \vee \phi_2, \alpha_1 \vee \alpha_2)$.

(AJ7) $JUSTIFY(t, \phi_1, \alpha_1) \wedge JUSTIFY(t, \phi_1 \rightarrow \phi_2, \alpha_2) \rightarrow JUSTIFY(t, \phi_2, \phi_1 \wedge (\phi_1 \rightarrow \phi_2))$.

(AJ8) $JUSTIFY(t, \phi, \alpha) \equiv JUSTIFY(t, JUSTIFY(t, \phi, \alpha), true)$.

(AJ9) $\neg JUSTIFY(t, \phi, \alpha) \equiv JUSTIFY(t, \neg JUSTIFY(t, \phi, \alpha), true)$.

(RJ2) From $\vdash \alpha \equiv \beta$ infer $\vdash JUSTIFY(t, \phi, \alpha) \equiv JUSTIFY(t, \phi, \beta)$.

(RJ3) From $\vdash \phi$ infer $\vdash \forall t(JUSTIFY(t, \phi, true))$.

Appendix - III: Foundational Theory of Belief Revision

Axiom of Foundational Expansion

(AFE1) $FEXP(t, \phi, u) \equiv (EXP(t, \phi, u) \wedge (BEL(u, \alpha) \rightarrow \exists\beta\ JBEL(u, \alpha, \beta))) \vee (EXP(t, \phi, v) \wedge ((BEL(v, \alpha) \wedge \neg\exists\beta\ JBEL(v, \alpha, \beta)) \rightarrow FCON(v, \alpha, u)))$.

(RFE1) From $\vdash \phi_1 \equiv \phi_2$ infer $\vdash FEXP(t, \phi_1, u) \equiv FEXP(t, \phi_2, u)$.

Axiom of Foundational Contraction

(AFC1) $FCON(t, \phi, u) \equiv (CON(t, \phi, u) \wedge (BEL(u, \alpha) \rightarrow \exists\beta\ JBEL(u, \alpha, \beta))) \vee (CON(t, \phi, v) \wedge ((BEL(v, \alpha) \wedge \neg\exists\beta\ JBEL(v, \alpha, \beta)) \rightarrow FCON(v, \alpha, u)))$.

(RFC1) From $\vdash \phi_1 \equiv \phi_2$ infer $\vdash FCON(t, \phi_1, u) \equiv FCON(t, \phi_2, u)$.

Axiom of Foundational Revision

(AFR1) $FREV(t, \phi, u) \equiv FCON(t, \neg\phi, v) \wedge FEXP(v, \phi, u)$.

Did Newton Solve the "Extended Prediction Problem"?

Manny Rayner
Swedish Institute of Computer Science
Box 1263
S-164 28 KISTA
Sweden

Abstract

The paper criticizes arguments recently advanced by Shoham and McDermott, which purport to establish the existence of the so-called "Extended Prediction Problem". We claim that the "problem" is the product of a mistaken understanding of the the formal basis of Newtonian mechanics, and has no real existence. An example is given showing how, contrary to Shoham and McDermott's arguments, it is possible to formalise reasoning about the evolution of physical systems in continuous time using only classical logic and differential calculus.

1 Introduction

The "Frame Problem" has now been around for a good while, but shows no sign of disappearing. In a recent paper, Yoav Shoham and Drew McDermott ([1988]; hereafter S&McD) give a new angle: they suggest dividing it up into two separate problems, which they refer to as the "qualification problem" and the "extended prediction problem". It is the second of these that will be my primary concern in the current document: I begin by summarizing S&McD's arguments.

S&McD's main claim is that there is a problem, the "extended prediction problem" (EPP). This is supposed to be the fact that it is difficult to formalize the process of making predictions over extended periods of time, if the axioms of a temporal theory are expressed as differential equations; S&McD claim moreover that the problem is not the product of a particular temporal formalism, but is general in nature. To substantiate this statement, they present arguments purporting to show that the problem occurs, not only in a conventional framework, but also in the Hayes "histories" formalism. In a later paper, Shoham [1988] then goes on to describe his logic of "Chronological Ignorance" (CI), which (among other things) is supposed to provide a solution to the EPP.

What is the actual problem supposed to be? S&McD are happy to agree that Newtonian mechanics is *in principle* capable of describing the behaviour of dynamic systems in continuous time (p. 52-53)[1]; however, they also claim that there is no well-defined associated *computational mechanism* which formalizes the process of making predictions. Considering the concrete problem of predicting the collision of two billiard balls, they write (p. 54):

> The "prediction," however, is purely model-theoretic. No attention was paid to the problem of actually *computing* the point of collision. In fact, it is very unclear how to perform the computation, since all axioms refer to time *points*. Somehow we must identify the "interesting" points in time or space, and interpolate between them. The problem seems a little circular, though, since the identity of the interesting points depends on the integration. For example, understanding where the two balls are heading logically precedes the identification of the collision: if we don't know that the two balls are rolling towards each other, there is no reason to expect a something interesting at the actual collision point.
>
> How do people solve such physics problems? The inevitable answer seems to be that they "visualize" the problem, identify a solution in some mysterious ("analog") way, and only then validate the solution through physics... (italics in original)

I will take the two paragraphs just quoted as the kernel of S&McD's claim. Before saying anything else, I think that it is important to point out that it is an extremely *strong* claim: all sorts of people are in the business of doing temporal reasoning using differential equations, and many of them would be prepared to defend themselves against the accusation that they are doing anything that couldn't be formalised. When the accusation is moreover exemplified in the trivial problem of predicting the collision of two billiard balls, the feeling that one is on theoretically secure ground is so strong as to more or less

[1] All page references are to [Shoham & McDermott 88] except where otherwise stated.

amount to certainty. Although I naturally don't mean by to imply that a feeling of certainty *proves* anything, this is worth saying, since it motivates most of the reasoning in the sequel. My counter-claim, then, will be that there is actually nothing mysterious about the process of making predictions about continuous-time processes, and that these can readily be formalized with no more theoretical apparatus than is afforded by classical logic, together with the differential and integral calculus.

The remainder of the paper is laid out as follows. In the next section, I will point out what I regard as several concrete logical errors in S&McD's analysis of the EPP, both in the classical and the "histories" frameworks; in section 3, I will go further to sketch how it is in fact perfectly possible to predict billiard-ball collisions, using only classical logic and well-defined and unmysterious methods of inference. In the last section, I sum up my conclusions.

2 Specific criticism of Shoham and McDermott's arguments

I will start with S&McD's treatment of the classical framework. Firstly, in several places in the argument, it certainly appears as though S&McD are committing the cardinal sin of confusing "infinitessimal" with "very small". Look, for example, at the following passage from page 59:

> The most conservative prediction refers to a very short interval of time, in fact an instantaneous one, but that makes it very hard to reason about more lengthy future periods. For example, if on the basis of observing a ball rolling we predict that it will roll just a little bit further, in order to predict that it will roll a long distance we must iterate this process many times (in fact, an infinite number of times). We will call this the *extended prediction problem.*

Now at risk of stating the obvious, it is not correct to say that a differential equation licences prediction "a little bit forwards", and then talk about doing this "many times - in fact an infinite number of times". Differential equations say things about the instantaneous rate of change of functions; to make predictions about extended periods they must be *integrated.* The integration will hold over a period if the differential equation holds over the same period, but the *length* of the period is completely irrelevant.

This is not mere pedantry; S&McD's lack of precision in expressing themselves is obscuring a crucial point. Since the differential equations don't directly allow forward prediction in the first place, the problem is not one of making an inefficient process more effective, predicting over a long interval rather than a short one. The problem is rather how we can justify prediction over any period at all. This is very much at odds with, for example, the following passage from p. 60 (my italics):

> To summarize, the general extended prediction problem is that although we may be able to make predictions about *short* future intervals, we might have to make a *whole lot of them* before we can predict anything about a substantial portion of the future.

All right, so why don't we just integrate the differential equations then? Now S&McD have another argument in reserve; as we have already seen in the passage quoted in section 1 above, they claim that we don't know what interval to integrate over. My second point is, very simply: *This is not a problem.* All that needs to be done is to perform the integration over an interval whose bounds are left unspecified, except by the restriction that the differential equation should hold within them; this is exactly what applied mathematicians normally do in practice. Call our bounds are t_1, t_2: then what we get is a logical formula of the form

conditions on what holds at t_1 *&*
the differential equations hold between t_1 *and* t_2 $\rightarrow$
conditions on what holds at each point between t_1 *and* t_2.

By using these formulas, together with other facts, we can deduce the maximal t_1 and t_2 over which the integration is valid. In the next section I will illustrate how this is done for the problem with the billiard balls.

My third point concerns the notion of "potential history", which is, I claim, a somewhat misleading concept. Instead of talking about "the way things would turn out if nothing happened" (Shoham's definition of a "potential history"), it will be quite enough to take "the way things *actually* turn out *until* something happens". Then it will be possible to reason that either

> i) nothing ever does "happen"
> ii) there is a first thing that "happens"

In case ii), we will be able to deduce things about when the aforementioned "first thing" occurs. If this sounds cryptic, the example in section 3 should make things clearer.

I now move on to the reformulation of the problem in Hayes's "histories" framework, dealt with by S&McD in their section 1.2. The logical fallacies here are of a similar type to those I have just pointed out, but since the integration has in effect already been performed they are of a more transparent nature.

In one sentence: the way in which S&McD use histories to express the problem isn't the right one. To back up this claim, let's start by reviewing the problem. The initial data is that there are two ROLLING histories, H11 and H21, of which we are given the prefixes, H11' and H21'. (See figures 1, 2 and 3, adapted from S&McD's diagrams 4, 5 and 6).

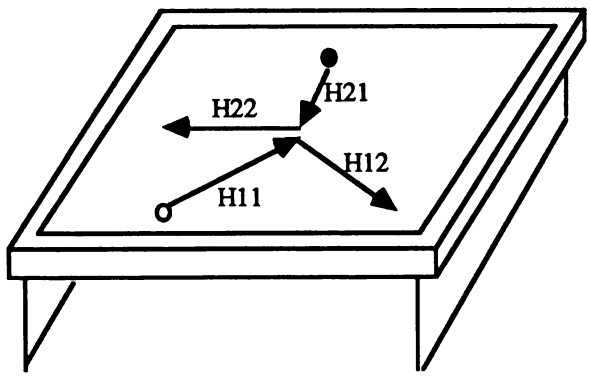


Figure 1

S&McD don't actually define exactly what they mean by a "prefix", but I suggest the following: it is the intersection of the history with some suitable given portion S of space-time. A simple way of defining S would be to let it be bounded above and below in time by two closely-spaced instants near the beginning of the period under consdieration. Anyway, S&McD now go on to say that we want to predict two "new" ROLLING histories. They then inquire what these new histories should look like: either they will extend up to the collision point, or they won't. I agree with their objection that the second alternative merely postpones the problem one step, but their analysis of the first alternative quite fails to hold water.

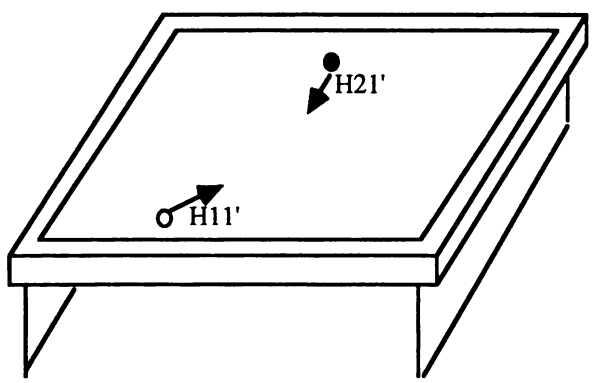


Figure 2

S&McD's point here is that what we want to say intuitively is that "the histories persist for as long as possible" - i.e. until they collide with something - and that "there only are two histories". They claim that there is a difficulty with the second part; that there are, actually, a lot more histories lying around, like for example the histories H11" and H21" which follow just after H11' and H21'. But this is just playing with words; obviously, every history contains an infinite number of subhistories, so counting all histories can never get us anywhere. What we are interested in are the *maximal* histories in a given bounded chunk of space-time, which in this domain *are* going to be finite in number. Now we can say that there are exactly two maximal histories in S (the chunk we intersected with in the last paragraph to define our prefixes); these are by construction H11' and H21'.

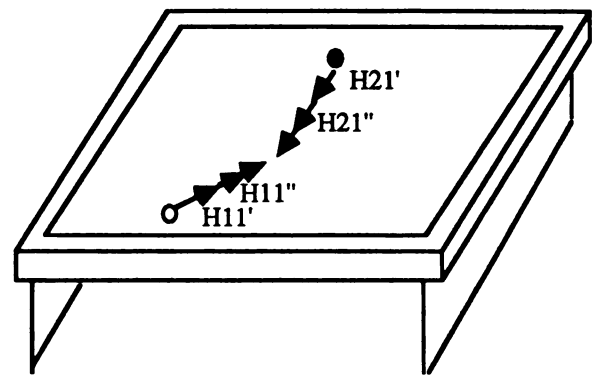


Figure 3

If we then move on to consider a larger chunk of space-time (call it S'), H11' and H21' are in general no longer going to be maximal. They will, however, be included in two unique maximal histories[1], which in a sufficiently large S' will be precisely H11 and H21. It is then fairly clear how to express our laws of physics so as to make things work. The rule we need is going to be something like the following:

Let t_1 and t_2 (with $t_1 < t_2$) be two times, and let S be the region of space-time bounded by t_1 and t_2. Assume that all the maximal histories in S are ROLLING histories, which touch both boundaries and not each other, and that there are exactly N such histories. If there are any collisions after t_2, call the earliest time at which one occurs T, and call the region on space-time bounded by t_1 and T, S'. Then there are exactly N+M maximal histories

[1] Proving the truth of this statement would demand a full axiomatization of the histories framework, something that is obviously impractical here. However, the key step would presumably be an axiom to the effect that the union of two connected histories of the same kind was a history.

in S', of which M are COLLISION histories and N are ROLLING histories. M < N, the intersection of each of the ROLLING histories with S is a distinct maximal history in S, and each of the COLLISION histories occurs at time T.

Together with the rule that COLLISION histories occur precisely when ROLLING histories meet, this will enable us to predict when the first collision occurs, using methods that essentially consist of little more than geometrical calculations in a three-dimensional Euclidean space.

3 Reasoning in continuous time: an example

Consider the situation illustrated in figure 4 below. (A propos S&McD's remarks above about "visualization" (p. 54): the diagram is purely for the benefit of human readers, and as will be seen contributes nothing to the proof). We have a billiard table, on which a system of Cartesian coordinates is defined. At time t=0 there are three balls: A at (0,2) with velocity vector (0,-2); B at (1,1) with velocity vector (0,-2); and C at (2,0) with velocity vector (-2,0). We will assume that balls are point objects which collide iff their positions coincide, and that the only forces on balls are those obtaining at times of collision. Given these assumptions, we want to know whether there will be any collisions, and if so when the first one will occur. Since we aren't going to reason past this point, we don't need to say anything about how balls behave after a collision, whether they bounce, stick together, smash or whatever. We reason as follows.

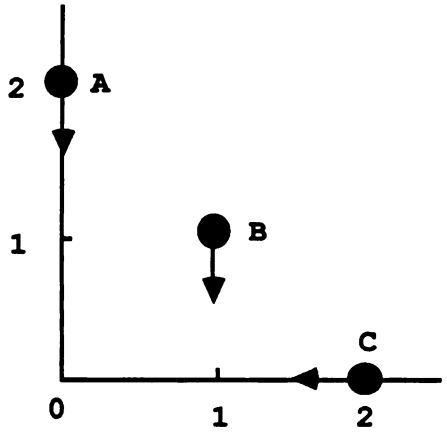


Figure 4

1) Either no collision will occur, or else there will be a first collision. In this case, one of the following will be true; A and B will be involved in it, B and C will be involved in it, or C and A will be involved in it. We refer to these four possibilities as No_collision, AB_first, BC_first and AC_first, and we refer to the time of the first collision as t_c with the convention that $t_c = \infty$ if No_collision obtains. (This is only to make the proof a little more elegant).

2) For $0 \le t < t_c$ no force acts on the balls. So by Newton's laws, their velocities will be constant during this period, and thus by performing an elementary integration we have that A's position at time t is (0,2-2t), B's is (1,1-2t) and C's is (2-2t,0), $0 \le t \le t_c$.

3) We want to prove that No_collision doesn't hold, so we use reduction ad absurdum and assume that it does. Thus the positions are as given by 2) for all positive t. We now want to prove that a collision *does* take place to get our contradiction; specifically, we try to prove that A will collide with C. This will be so if we can find a positive t such that

$$(0,2-2t) = (2-2t,0)$$

Elementary algebra shows that t = 1 is a (in fact, the only) solution. So No_collision doesn't hold.

4) We now prove that AB_first doesn't hold. Again, suppose it did. Then the moment of collision is given by the equation

$$(0,2-2t_c) = (1,1-2t_c)$$

This gives us that 0 = 1, a contradiction. So AB_first doesn't hold either.

5) Now prove that AC_first doesn't hold. Once more, suppose it did. Then just as above, we have that t_c is given by

$$(0,2-2t_c) = (2-2t_c,0)$$

Algebra gives us that $t_c = 1$. We must now establish that some other collision occurred at some earlier time, to obtain our contradiction. Specifically, we try to prove that B will collide with C. This will be so if we can find a t' such that

$$(1,1-2t') = (2-2t',0)$$

and $0 \le t' < 1$. But t = 0.5 is such a solution, and once again we have a contradiction.

6) We have proved that there will be a collision, so there must be a first collision. Since AB_first and AC_first have been proved impossible, by elimination we have BC_first. Doing the same bit of algebra as in 5) shows that it occurs at t = 0.5. QED

Let us consider the structure of this proof. First, in 1), we hypothesize a time-point t_C, which is when the first thing is going to "happen". The important thing to notice here is that we don't yet say what t_C's *value* is; we define it in terms of its properties, namely that a collision occurs then, and that none occur before. Then in 2) we integrate the differential equations to get the interval-based information that will allow us to make predictions. If we had been working in a histories formalism, the equations would as indicated earlier "already be integrated", and this step would have been superfluous. Having got this far, the remaining steps 3) to 6) are just ordinary monotonic classical logic, and consist of a proof that t_C as described actually does exist, together with a computation of its value. It is clear that the methods used are quite general, and in no way make special use of the billiard-ball scenario. One incidental point is also worth noting explicitly: the collision is shown to have occurred at t=0.5, showing that we really have moved outside the integers.

To point the moral, the proof above demonstrates that classical logic and differential calculus are at any rate *sufficient* to solve problems of this kind. Shoham (personal communication) has however advanced another criticism: he claims that, although the approach I have just demonstrated is possible, it is less *efficient* than using CI.

My answer to Shoham's objection is twofold. Firstly, this is not what is being said in the original paper; it is a separate issue, which as far as I can see is nothing to do with the EPP. Secondly, Shoham has still to demonstrate that proofs of this kind can be carried out *at all* in CI. As he admits himself ([Shoham 1988], p. 320), the methods he has so far developed are completely dependant on the use of a discrete model of time; doing CI in continuous time would require the use of different algorithms, the complexity of which is thus completely unknown. Shoham might be right, but he has to present evidence to prove it; to use a metaphor from another game, the ball is now back in his court.

4 Conclusions

What I have shown in this paper should not in any way be regarded as startling or unexpected; it is simply the defence of what most mathematicians and physicists would unhesitatingly call the common-sense view, namely that there is no longer anything mysterious about the EPP. If we take a broader historical perspective, however, we can see that the EPP *used* to be a major problem. It is in fact closely related to Zeno's paradox, something that caused philosophers difficulties from Zeno's time until the seventeenth century; until then, nobody even came close to explaining how it was possible to use logic to reason rigorously about continuous change. Indeed, many prominent thinkers went on record as claiming that such things were impossible in principle.

The first person to give a plausible account of mathematical reasoning about continuous processes was Newton, and even *he* was unable to do this in a satisfactorily formal way; this was not achieved until the nineteenth century analysts - people like Bolzano, Dedekind, Riemann and Cauchy - finally managed to put real analysis onto a sound logical footing. Not being an expert on the history of mathematics, I can't say with confidence just when the whole enterprise was completed; but I would be prepared to guess that Russell and Whitehead still had to tie up a few loose ends in the *Principia Mathematica*. The whole process, in other words, took over two hundred years.

To sum up, then, the EPP is undoubtedly an extremely important and difficult problem. It is, however, a problem that has been *solved*; to suggest otherwise is only to mock what must surely rank as one of the greatest achievements in the history of science and mathematics.

Acknowledgements

I would very much like to thank Yoav Shoham for his unfailingly courteous and helpful attitude in the face of hostile criticism. The comparison between the EPP and Zeno's paradox in the last section is due to Rune Gustavsson, and the proof in section 3 benefitted considerably from two suggestions made by Annika Wærn.

References

[Shoham & McDermott 88] Yoav Shoham & Drew McDermott. Problems in Formal Temporal Reasoning. *Artificial Intelligence*, 36:49-61, 1988

[Shoham 88] Yoav Shoham. Chronological Ignorance: Experiments in Nonmonotonic Temporal Reasoning. *Artificial Intelligence*, 36:279-331, 1988

Synthesizing Information-Tracking Automata from Environment Descriptions*

Stanley J. Rosenschein
Teleos Research
576 Middlefield Road
Palo Alto, CA 94301

Abstract

This paper explores the synthesis of finite automata that dynamically track conditions in their environment. We propose an approach in which a description of the automaton is derived automatically from a high-level declarative specification of the automaton's environment and the conditions to be tracked. The output of the synthesis process is the description of a sequential circuit that at each clock cycle updates the automaton's internal state in constant time, preserving as an invariant the correspondence between the state of the machine and conditions in the environment. The proposed approach allows much of the expressive power of declarative programming to be retained while insuring the reactivity of the run-time system.

1 Introduction

This paper is concerned with the synthesis of finite automata whose internal states are provably correlated with changing conditions in the environment. In earlier work [Rosenschein, 1985, Rosenschein and Kaelbling, 1986], we investigated the mathematical foundations of embedded machines and direct methods of programming them. Later research was aimed at raising the conceptual level of the programming task by exploring declarative techniques for synthesizing their action-selection circuitry [Kaelbling, 1988]. In this paper, we extend this line of research to perceptual updates, that is, the computations responsible for updating the internal information state of the machine. We present techniques that allow programmers to describe the environment in which a machine is to be embedded along with conditions to be tracked and to have these descriptions algorithmically transformed into provably real-time circuitry for tracking those conditions in the environment. Information about these conditions would be used by other parts of the system to guide action.

Mainstream theoretical AI has developed models of information and action based on formalized commonsense psychology. In this approach, intelligent computer systems are modeled as having at their disposal a set of propositional "beliefs," usually assumed to be embodied in a set of symbolic expressions, such as logical formulas, whose intended semantics are clear to the designer. Some of these beliefs are provided by the designer as part of a knowledge base, while others are produced by the perceptual system at run time. In addition, the system contains inference procedures for dynamically deriving new beliefs from old and for continuously revising beliefs over time in response to sensory inputs (and perhaps reflection.) In this way, the designer can arrange for the agent to have access to a much more complex set of beliefs than could have been enumerated explicitly in advance. The designer also provides symbolic representations of the goals the agent is to pursue. The agent continuously attempts to deduce which actions it should take to achieve its goals and then performs those actions.

By modeling the information available to the system as symbolic facts deducible by the system, the traditional approach allows the methods of symbolic logic, including automated symbolic inference, to be applied to problems in agent design. Of particular importance is the availability of a clear semantics for non-numerical data structures that are used to represent qualitative information about the world. These are attractive features—ones we would like to preserve. However, the traditional AI approach also has some other, less attractive, features which we hope to eliminate. For example, in applications requiring continuous, high-speed interaction with the environment, the computational cost of formally deriving facts from a data base of logical premises and of keeping the data base consistent with the world is often prohibitive. This has been a severe obstacle to building high-performance embedded computer systems based on the model of the intelligent agent as symbolic reasoner.

Situated-automata theory is a framework for rec-

*This work was supported in part by a gift from the System Development Foundation.

onciling the attractive features of AI methods (nonnumerical descriptions of the world) and of control-theoretic methods (continuous constant-time updating of internal representations and guaranteed response.) The central observation of situated-automata theory is this: It is not the run-time symbols or numbers, per se, that are of significance, it is the fact that (1) they are semantically meaningful to the designer, that is, they stand for well-defined world conditions, and (2) the machine is designed in such a way that the world condition represented by the value of an internal location will indeed hold when that location has that value.

In this paper, we apply the situated-automata framework to the problem of synthesizing machines that track semantically complex conditions in the environment using constant-time update circuitry. We describe how inference techniques can be used at compile time to carry out the synthesis automatically, given symbolic descriptions of the environment and of the information to be tracked.

2 Basic concepts: A model of information

The mathematical framework of situated-automata theory takes as its starting point a model of dynamic systems. Consider a physical or computational system consisting of a set of locations that can be in different states over time. These states can be thought of as actual physical states or as abstract data values that might be stored in the register of a computer. Let T be a set of times, L a set of locations, and let each location a take on values from some set, D_a, with compound locations $[a,b]$ taking on values in $D_a \times D_b$. Let the union of all value domains be designated by D. Each possible "trajectory" of values can be given by a function $w : L \times T \rightarrow D$, in which $w(a,t)$ is the value of location a at time t in trajectory w. Following the terminology of possible-worlds semantics, we call these trajectories "worlds."

In physical or computational systems that operate according to fixed rules or constraints, not every world is consistent with the laws of nature. This can be captured mathematically by identifying some designated subset of worlds that are consistent with the intended constraints. We shall call this set of possible worlds W. Let Φ, the set of *propositions* or *world conditions*, be the set of all subsets of $W \times T$. Intuitively, a condition $\varphi \in \Phi$ corresponds to the set of world-time points at which that condition holds. We sometimes write $\varphi(w,t)$ rather than $\langle w,t\rangle \in \varphi$ when we wish to assert that the condition φ holds at w,t.

By definition, Φ has the structure of a Boolean algebra of sets: a condition φ can *imply* (be a subset of) another condition ψ, we can take the *meet*, $\varphi \cap \psi$, of two conditions, and so on. Furthermore, the structure of Φ allows us to define two mathematical objects useful for characterizing world dynamics: the *initial condition* $\varphi_0 = \{\langle w,0\rangle \mid w \in W\}$ and the *strongest postcondition function* $S : \Phi \rightarrow \Phi$, where $S(\varphi) = \{\langle w,t+1\rangle \mid \varphi(w,t)\}$. The initial condition φ_0 and the strongest postcondition function S will be used later to characterize machine synthesis.

The restriction on what is possible gives rise directly to a notion of information. The information contained in a location's value is modeled as the strongest proposition consistent with that location's having that value. Formally stated, to every location (or compound location) a, we associate a function, $M_a : D_a \rightarrow \Phi$ that maps a's values into propositions. This function is defined as follows: $M_a(v) = \{\langle w,t\rangle \mid w(a,t) = v\}$. To say that a location a has the information that φ in world w at time t is to say $M_a(v)$ implies φ, in other words, that the proposition φ is true at each world and time in which a has the same value it has in world w at time t.

As defined, the concept of information is very abstract, representing the totality of what must be the case, given that some location in the machine has the value it does. For this notion to be of practical use, we must find ways of expressing in understandable terms particular, more limited, aspects of this total information content. This is the proper role of logic. By defining logical languages whose formulas express propositions of interest, we can conveniently describe the content of propositions included in an agent's information state, such as $in(book, room1) \vee in(book, room2)$. Furthermore, modal logics of knowledge can be used to assert facts about the information relation itself, such as whether particular locations have or do not have particular information, e.g., $\neg K(a, in(book, room2)) \wedge K(b, in(book, room2))$, which asserts that location a does not contain the information that the book is in room 2, while location b does. These logics are explored more fully in [Rosenschein and Kaelbling, 1986] and [Halpern and Moses, 1985]. In this paper, we will use letters $p, q, \ldots$ and standard logical operators $\wedge, \vee, \ldots$ in formulas that express the information carried in a location's value.

When we wish to consider machines with very large state sets, we regard the machine as being constructed from a network of components, with the state set of the whole machine corresponding to the Cartesian product of the state set of the individual components. Fortunately, there are straightforward techniques for inferring informational properties of aggregates from informational properties of their components. For instance, the following can easily be shown to be valid:

$$M_{[a,b]}([u,v]) = M_a(u) \cap M_b(v)$$

We refer to this property as *spatial monotonicity*. It follows that if location a carries the information that p holds and location b carries the information that q holds, then the aggregate location $[a,b]$ carries the in-

formation that the conjunctive condition $p \wedge q$ holds. Spatial monotonicity is useful in synthesis because it means that subsystems can be developed independently and composed in a meaningful way.

It is important to observe that location a can carry information about p without explicitly encoding a symbolic formula representing p; any value of the location that is systematically correlated with p will suffice. Different locations might have different states representing the same proposition p, and the same data values might have different informational significance at different locations. In general, an infinite number of formulas will follow from the information contained in a finite value, but since the formulas need not be separately represented, this causes no problem. Indeed, the computational complexity of updating a location's value so that it tracks changing conditions in the environment is entirely decoupled from the computational complexity of the symbolic inference problem for the logical language that expresses the conditions being tracked. This fact is crucial for understanding how seemingly complex semantic conditions can be monitored in real time.

These considerations lead directly to certain abstractions useful for describing how information is represented in machine states and how it is re-represented as it moves from location to location within a machine. We call these abstractions *informational data types.*

3 Informational data types

Recall that the function M_a maps a's values (elements of D_a) to the abstract propositions with which they are correlated. As such, we can think of it as a "meaning" function. It is also useful to consider an inverse to these meaning functions, namely, "representation" functions that map propositions back to the data values that encode them. Let us define an *informational data type* to be a triple $\tau = \langle D, M, R \rangle$, with D being a set of data values, M a meaning function from D to Φ, and R a representation function from Φ to D. Intuitively, if a location is of type τ, then whenever it takes on the value $v \in D$, the world is intended to satisfy condition $M(v)$, and if the world satisfies condition φ, it is appropriate for the location to take on the value $R(\varphi)$. These two functions must satisfy the property that φ implies $M(R(\varphi))$ for all $\varphi \in \Phi$, that is, the representation map must be consistent with the meaning map. Since the implication is only one way, the representation of a world condition in the machine will, in general, not be information preserving. An extreme case of this is when $M(R(\varphi)) = \mathit{true}$ and contains no information at all. We generally assume $R(\varphi)$ to be maximally specific, that is if $R(\varphi) = v_i$, then there is no v_j such that $M(v_j)$ implies, but is not implied by, $M(v_i)$.

Informational data types provide a way of analyzing the localization of information in a machine, including the computational complexity of such localization. Given two informational data types, $\tau_1 = \langle D_1, M_1, R_1 \rangle$ and $\tau_2 = \langle D_2, M_2, R_2 \rangle$, we can define a translation function that "re-represents" the content implicit in the values of a location of type τ_1 in the language of τ_2. Mathematically, the translation function is a mapping $T_2^1 : D_1 \rightarrow D_2$ that is defined as follows: $T_2^1(v) = R_2(M_1(v))$, i.e., the representation, in the second "language," of the meaning, in the first "language," of v. The computability and complexity of these translations remain to be determined and are greatly affected by the choice of representation, that is, by the specific nature of M and R and not merely by the range of propositions encoded.

Although the informational concepts developed thus far apply equally to finite and infinite languages, we shall henceforth restrict our attention to machines having a finite number of internal locations, each taking on values from a finite domain. One immediate consequence is that all translation functions are computable, although complexity trade-offs remain. For instance, since all Boolean functions can be computed by a circuit of depth 2, we could always compute the translation function in constant time—if we were prepared to tolerate the potentially large number of computing elements that may be required. In the worst case, an exponential number of gates could be needed and the constant-time result is merely academic. Our aim, however, is to control the synthesis process in order to produce systems that not only track world conditions but are also practical to design and implement.

In the next section we discuss how informational data types can be used to approach the synthesis problem.

4 From analysis to synthesis: The machine induced by world dynamics

One way of using the situated-automata framework is for the analysis of existing machines: Given the description of an environment and of a machine embedded in that environment, we seek to describe the information encoded in its states. For purposes of design, however, we are more interested in the opposite question: Given a description of the world and of the information we would *like* to have encoded in machine states, how can we design the machine's circuitry in such a way that states of the machine will actually be correlated, as desired, with conditions in the world?

At the theoretical level, we can show that the dynamics of the world, together with the semantics of the machine's inputs and the *intended* semantics of its internal states (expressed as an informational data type), fully determine a machine whose internal states carry the desired information by virtue of their actual correlation with the world. To see why this is the case, imagine we are given an informational data

type $\tau_{in} = \langle D_{in}, M_{in}, R_{in} \rangle$ for the input location of a machine and an intended data type $\tau_a = \langle D_a, M_a, R_a \rangle$ for the internal location a. Imagine, further, that we are given the proposition φ_0 approximating the initial world condition (in the sense that φ_0 is implied by the true initial condition), and a function S approximating the true strongest-postcondition function (in the sense that $S(\varphi)$ is implied by the true strongest postcondition of φ for each φ; approximation is the best we can do, since the world is not fully determined until we have fixed the embedded automaton.) We now show how these elements determine a machine that tracks changing world conditions as desired.

Here a machine will be defined by a pair of domains D_{in} and D_a (for inputs and internal states), an initial value $v_0 \in D_a$, and a next-state function $f : D_{in} \times D_a \rightarrow D_a$ satisfying the following conditions for all w, t:

$$\begin{aligned} w(a,0) &= v_0 \\ w(a,t+1) &= f(w(in,t), w(a,t)) \end{aligned}$$

Given τ_{in}, τ_a, φ_0, and S, we define v_0 and f as follows:

$$\begin{aligned} v_0 &= R_a(\varphi_0) \\ f(u,v) &= R_a(S(M_{in}(u) \cap M_a(v))) \end{aligned}$$

Intuitively, v_0, the initial value of the location a, is just the representation, in a's data type, of the initial proposition; the value of the next-state function is determined mathematically by considering the proposition associated with the old value of a and with the input, determining what will be true one time instant in the future given what is true now, and representing that proposition in the data type of a. Note the implicit reliance on spatial monotonicity and the similarity between this construction and the definition of translation functions in the previous section.

Assuming M_{in} is veridical, it can be shown that these definitions of v_0 and f insure that M_a will be veridical as well, i.e., that the machine's states will indeed be correlated with the intended meanings of those states. Mathematically, we must demonstrate that for all w, t:

$$M_a(w(a,t))(w,t).$$

The proof of this proposition is straightforward and proceeds by induction on t. (The variable w is universally quantified throughout.) The base case is established as follows: From the definition of φ_0, we have that $\varphi_0(w,0)$. The soundness of R_a gives us $M_a(R_a(\varphi_0))(w,0)$, whence $M_a(w(a,0))(w,0)$ follows immediately from the definition of v_0 by simple substitution, since $w(a,0) = v$.

The inductive case is established similarly. The induction hypothesis is that

$$M_a(w(a,t))(w,t),$$

and the veridicality of M_{in} gives us

$$M_{in}(w(in,t))(w,t).$$

Conjoining these conditions yields

$$(M_{in}(w(in,t)) \cap M_a(w(a,t)))(w,t).$$

The definition of S implies that

$$S(M_{in}(w(in,t)) \cap M_a(w(a,t)))(w,t+1),$$

and the soundness of R_a guarantees that

$$M_a(R_a(S(M_{in}(w(in,t)) \cap M_a(w(a,t)))))(w,t+1).$$

Substituting in the definition of f, we get

$$M_a(f(w(in,t), w(a,t)))(w,t+1),$$

from which $M_a(w(a,t+1))(w,t+1)$ follows immediately.

5 Syntactifying the construction

We have just seen how the dynamics of the environment, together with the semantics of the inputs and the intended semantics of the internal state, completely determine the structure of a machine. To be of practical utility, however, the mathematical construction must be made operational. One approach would be for the programmer, based on his intuitive understanding of the task environment, to define the induced automaton directly in a conventional programming language. Although adequate in principle, this approach is difficult to apply in practice for complex domains. For this reason we seek compilation techniques that would automate at least part of the synthesis process and make the transition from environment description to automaton more transparent to the programmer.

Although the automaton is mathematically determined by τ_{in}, τ_a, φ_0, and S, we cannot directly present these abstract objects to a compiler and must use symbolic, often approximate, descriptions. Let us examine the form these descriptions might take for informational data types (τ_{in} and τ_a) and for world dynamics (φ_0 and S).

5.1 Specifying informational data types

Consider a machine location x of informational data type $\tau_x = \langle D_x, M_x, R_x \rangle$. Let us see how the three components of the data type might be described to a compiler.

The value domain D_x is straightforward to describe using conventional data type declarations. For our purposes, it will be sufficient to consider only atomic data types such as Booleans, integers, floats, etc. and record structures, possibly nested, over these atomic types. For example, to tell the compiler about the value domain of location x we might write `x: [bool [int int] float]`.

The specification of the other two components is more complex. Let us begin with M_x. Recall that M_x, the "meaning function" associated with location x, maps elements of D_x to Φ, the set of propositions,

where propositions are modeled as sets of world-time pairs. Recall also that logical formulas can be used to partially express the content of a proposition, provided they are taken from a logic that assigns sets of world-time pairs as the denotation of formulas. Many temporal logics will suffice for this purpose. (For one example, see [Rosenschein and Kaelbling, 1986].) Given such a logic, formulas parametrized by run-time values can be used to define a meaning function from values to propositions.

To see this, let $\mathcal{L}$ be the language of some temporal logic, with interpretation function $\mathcal{I} : \mathcal{L} \rightarrow \Phi$ and provability relation $\vdash$. Assume that among its individual terms, the language has constants that rigidly designate values of locations, possibly in addition to terms that denote location values that vary with world and time. If v is a data value, we let c_v stand for the rigid designator of value v in the language $\mathcal{L}$.

Let $P_x(U)$ be a formula of $\mathcal{L}$ with a free variable U for which value-denoting terms can be substituted. Each substitution instance $P_x(c_v)$ is a closed formula to which the interpretation function $\mathcal{I}$ can be applied. The parametric formula $P_x(U)$ thus induces a mapping $\hat{M}_x : D_x \rightarrow \Phi$ that approximates M_x and is defined as follows:

$$\hat{M}_x(v) = \mathcal{I}(P_x(c_v)) .$$

The semantic interpretation $\mathcal{I}$ of the language $\mathcal{L}$ is itself approximated for the compiler by a set, Γ, of background facts relative to which the syntactic consequences of $P_x(c_v)$ are to be derived. To answer the question "does $M_x(u)$ imply $M_y(v)$," the compiler would attempt to establish $\Gamma \vdash P_x(c_u) \rightarrow P_y(c_v)$ using deductive techniques.

Having approximated M_x by a formula $P_x(U)$ and a background theory Γ, we have nearly determined the third component of the informational data type as well. Recall that the representation function R_x is intended to map propositions to elements of the value domain that best capture them. Since we are encoding propositions for the compiler using formulas, we are interested in functions that map formulas to data values. If Q is a formula expressing the proposition φ, candidate representations of φ (relative to Γ) should be drawn from the set

$$C_x(Q) = \{v \mid \Gamma \vdash Q \rightarrow P_x(c_v)\}.$$

Intuitively, elements of $C_x(Q)$ are the data values whose meaning is entailed by the information in Q, and hence by φ. This set must contain at least one element, since there must be a value in D_x representing *true*, which is entailed by every proposition. In practice, it is convenient to have multiple representations of *true*, for instance by uniformly including a boolean *valid bit* in all data types. When the valid bit has the value zero, the meaning of the whole value is taken to be simply *true*, regardless of the values of the rest of the parameters.

If there is more than one element in $C_x(Q)$, the multiple values must somehow be combined so that they might "fit" into the space alloted to x. We call the functions responsible for combining these values *rconj* functions ("representation of the conjunction.") This function is defined as

$$rconj_x(v_1, v_2) = R_x(M_x(v_1) \cap M_x(v_2)).$$

Because the choice of R_x, and hence *rconj*, is underdetermined by the meaning function M_x, the designer must somehow stipulate the $rconj_x$ function directly for each type τ_x. This can be made relatively convenient through the use of declarative rules of the form

$$P_x(V_1) \wedge P_x(V_2) \rightarrow P_x(f(V_1, V_2)).$$

Having specified a binary *rconj* operator, arbitrary finite sets of values can be combined in the obvious way:

$$rconj^*_x(v_1, \ldots, v_n) = rconj_x(v_1, \ldots rconj_x(v_{n-1}, v_n) \ldots)$$

The order of combination does not matter since the *rconj* function is commutative due to the commutativity of the underlying conjunction operator $\cap$ in terms of which *rconj* is defined.

Example

We illustrate the concepts above by defining a sample informational data type. Consider a location x of value type `[bool int]`. Informally, the first field is the valid bit, and the second field is intended to represent a lower bound on the age of some individual, Fred.

The semantics of x can be expressed using the formula $P_x(U) = age(fred, [first(U), second(U)])$ under the intended interpretation:

$$\mathcal{I}(age(fred, [V_1, V_2])) = \begin{cases} true & \text{if } V_1 = 0 \\ \varphi(V_2) & \text{if } V_1 = 1 \end{cases}$$

where $\varphi(V_2) = \{w, t \mid age(fred, w, t) \geq$ *the number denoted by* $V_2\}$. Thus, the run-time value $[0, n]$ at location x would represent the vacuous proposition *true* for any n, the value $[1, 14]$ would represent that Fred is at least 14 years old, and so on.

An *rconj* rule for the informational data type might look like this:

$$age(fred, [U_1, U_2]) \wedge age(fred, [V_1, V_2]) \rightarrow age(fred, [or(U_1, V_1), if(U_1, if(V_1, max(U_2, V_2), U_2), V_2)]).$$

This rule indeed defines a commutative *rconj* operator that can be used to combine values of this informational data type in a way consistent with the intended interpretation.

Furthermore, if the intended model incorporates the constraint that 18-year olds can vote, we might include among the background facts an assertion of the form

$$age(fred, [U_1, U_2]) \rightarrow$$
$$can\text{-}vote(fred, and(U_1, ge(U_2, 18))).$$

This fact implicitly defines part of a translation function from the *age(fred,—)* data type to the *can-vote(fred,—)* data type. Notice that in this data type, *fred* is fixed at compile time. This would be appropriate if distinct locations were used to store information about individuals referred to explicitly at compile time. If Fred's identity were not known until run time, the run-time parameter would have to take on values that encoded propositions about Fred, Sam, etc.

5.2 Specifying World Dynamics

As we described above, the compiler is assumed to have access to a background theory, that is, a set of assertions describing the environment. This theory will contain many temporal facts as well as atemporal facts. By choosing the language $\mathcal{L}$ to include appropriate temporal operators we can express facts about the initial condition of the world and about temporal transitions in a way that allows us to approximate the semantic objects φ_0 (initial condition) and S (strongest postcondition function.)

In the simplest case,the language $\mathcal{L}$ need only include the modal operators $\Box$, *init*, and *next* satisfying the following semantic properties for all w, t:

$$w,t \models \Box\varphi \text{ iff } w',t' \models \varphi \text{ for all } w',t'$$
$$w,t \models init\ \varphi \text{ iff } w,0 \models \varphi$$
$$w,t \models next\ \varphi \text{ iff } w,t+1 \models \varphi$$

The compiler can answer questions of the form "does $S(M_x(u))$ imply $M_y(v)$" by establishing $\Gamma \vdash \Box(P_x(c_u) \rightarrow next\ P_y(c_v))$ using deductive techniques.

5.3 Synthesis Method

We now describe a compilation method that operates on the representations discussed above and produces a circuit description of the desired automaton. The compiler takes as inputs a description of information carried by the run-time inputs to the machine and the internal machine state, as well as a background theory containing temporal facts. The compiler operates by deriving theorems about what is true initially and about what will be true next at any time, given what is true at that time. In the course of the derivation, free variables are instantiated in the manner of logic programming systems. From the instantiated formulas the compiler extracts the initial value of the machine's internal state and the description of a circuit for updating the machine's state vector.

More precisely, the compiler's inputs consist of the following:

- a list $[a_1, ..., a_n]$ of input locations
- a list $[b_1, ..., b_m]$ of internal locations
- for each input location a, a formula $P_a(U)$ with free variable U
- for each internal location b, a formula $P_b(U)$ with free variable U and a function $rconj_b$
- a finite set Γ of facts.

For each internal location b, the compiler computes two sets of value terms I_b and N_b defined as follows:

$$I_b = \{e \mid \Gamma \vdash \Box init\ P_b(e)\}$$
$$N_b = \{e \mid \Gamma \vdash \Box(P_{a_1}(a_1) \wedge \ldots \wedge P_{b_n}(b_n) \rightarrow next\ P_b(e))\}.$$

If these sets are infinite, they can be generated and used incrementally. This is discussed more fully below.

From these collections of sets the compiler computes the initial value and the update function. The initial value is computed as follows:

$$v_0 = [rconj^*_{b_1}(I_{b_1}), \ldots, rconj^*_{b_n}(I_{b_n})],$$

In other words, the initial value of the state vector is the the vector of values derived by *rconj*-ing values representing the strongest propositions that can be inferred by the compiler about the initial state of the environment in the "language" of each of the state components. Similarly, for the next-state function:

$$f([a_1, \ldots, a_n], [b_1, \ldots, b_m]) =$$
$$[rconj^*_{b_1}(N_{b_1}), \ldots, rconj^*_{b_n}(N_{b_n})].$$

Here the compiler constructs a vector of expressions that denote the strongest propositions about what will be true next, again in the language of the state components.

In the case of the initial value, since all the terms are rigid, the *rconj* values can be computed at compile time. In the case of the next-state function, however, the *rconj* terms will not denote values known at compile time. Rather, they will generally be nested expressions containing operators that will be used to compute values at run time. Assuming the execution time of these operators is bounded, the depth of the expressions will provide a bound on the update time of the state vector.

Without restricting the background theory, we cannot guarantee that the sets I_b and N_b will be finite. However, even in the unrestricted case the finiteness of terms in the language guarantees that whichever elements we *can* derive at compile time can be computed in bounded time at run time. Furthermore, the synthesis procedure exhibits strongly monotonic behavior: the more elements of I_b and N_b we compute, the more information we can ascribe to run-time locations regarding the environment. This allows incremental improvements to be achieved simply by running the compiler longer; stopping the procedure at any stage will still yield a correct automaton, although not necessarily the automaton attuned to the most specific information available. Since, in general, additional

rconj operations consume run-time resources, one reasonable approach would be to have the compiler keep track of run-time resources consumed and halt when some resource limit is reached.

As we have observed, without placing restrictions on the symbolic language used to specify the background theory, the synthesis method described above would hardly be practical; it is obvious that environment-description languages exist that make the synthesis problem not only intractable but undecidable. However, as with Gapps [Kaelbling, 1988] and other formalisms in the logic programming style, by restricting ourselves to certain stylized languages, practical synthesis techniques can be developed.

We have experimented with a restriction of the logical language that seems to offer a good compromise between expressiveness and tractability. This restriction is to a weak temporal Horn-clause language resembling Prolog but with the addition of *init* and *next* operators. In this language the background theory is given as facts of the following form:

```
init q(X,Y).
next q(X,f(X,Y)) :- q1(X,Y),...,qn(X,Y).
q(X,f(X,Y)) :- q1(X,Y),...,qk(X,Y).
```

For each predicate or function expression, the first argument represents a compile-time term and the second a run-time term. Facts of the first two sorts assert temporal facts (the $\Box$ operator is implicit), and facts of the last sort are ordinary instantaneous facts, much as one would find in a conventional Prolog system, but with terms syntactically marked as compile-time or run-time. The *rconj* rules are given in the following form:

```
rconj q(x,f(x,y1,y2)) :- q(x,y1), q(x,y2).
```

The derivation process proceeds as described above but uses backward-chaining deduction techniques adapted from logic programming. Each distinguished location `i` has an associated atomic formula schema `p(i,Y)`. In deriving the initial value v_0 the compiler attempts to prove `p(i,Y)` from the `init` declarations and the instantaneous facts. If this succeeds, the initial value of the `i`'th component of the state vector is the *rconj** of the bindings of `Y`. If the attempt fails, the valid bit of that component is set initially to 0. Similarly, the next-state function for component i is derived by attempting to prove `next p(i,Y)` using the `next` rules and the instantaneous facts, chaining backwards and cutting off proofs that traverse more than one "`next`" clause. Although the process of finding the proofs need not be real-time, the circuit that is finally produced is.

A prototype system, called RULER, has been built implementing the Horn-clause version of the synthesis algorithm. The language resembles Prolog in many ways, differing mainly in the strong distinction between compile-time and run-time expressions. Compile-time expressions undergo unification in the ordinary manner; run-time expressions, by contrast, are simply accumulated and used to generate the circuit description. RULER was implemented in Lisp as an extension of the Rex language [Kaelbling, 1987]. Run-time expressions in RULER are allowed to be any valid Rex expression, and all of the Rex optimizations (common-subexpression eliminiation, constant folding, etc.) are applied to the resulting circuit desciptions produced by RULER. The RULER system was run on several small examples involving object tracking and aggregation, and the synthesis procedure has proved tractable in our test implementation.

6 Future Directions

Our current research is directed toward extending the theoretical basis for synthesis and improving the practical utility of tools such as RULER. On the theoretical side, one important extension is to adapt the synthesis techniques to cases where the correlation between machine states and world conditions is best described probabilistically. Naive approaches will not work, primarily because the spatial-monotonicity property fails in the probabilistic case. For this reason we have been exploring design disciplines that reconcile structured synthesis methods with the inherently non-monotonic nature of probabilities, preserving the spirit of the techniques presented in this paper.

On the practical side, experiments with RULER suggest needed improvements in several areas. One syntactic improvement would be to uniformly suppress valid bits, since their treatment is so systematic. Freer syntactic intermingling of compile-time and run-time expressions and tests would be useful as well. A more serious practical consideration has to do with helping the programmer control the combinatorics of the synthesis process. As in general logic programming, it is possible, using RULER, to write programs with unacceptable combinatorial behavior. While this is not the fault of RULER per se and can undoubtedly be ameliorated by increasing the programmer's experience and skill in using the tool, there are improvements that can be made in the system itself, including facilities for detecting cycles and redundant proofs. Finally, there is need to gain practical experience in applying this style of programming to real problems in visual perception, sensor fusion, and other similar areas.

7 Related Work

There has been considerable work on the synthesis of digital machines from temporal logic specifications, for example, the work by Ben Moskowski [Moszkowski, 1983]. This work considers symbolic specifications similar to the kind considered here but does not connect them directly to an informational account of machine states. The work of Joseph Halpern and his

asssociates [Halpern and Moses, 1985], on the other hand, has examined mathematical approaches closely related to our own for characterizing information in distributed system, but have so far not addressed issues of automated synthesis. Chris Goad [Goad, 1986] has used partial evaluation to generate efficient algorithms for visual recognition. Goad's techniques are rather domain-specific and do not handle tracking of conditions over time. There is a rich literature in the traditional AI paradigm (as well as in formal philosophy) on belief revision (see, for example [Doyle, 1979, de Kleer, 1986]), but little work has addressed the implications of real-time update requirements.

Acknowledgments

I have benefited greatly from discussions with Leslie Kaelbling, Michal Irani, and David Chapman.

References

[Rosenschein, 1985] Rosenschein, Stanley J. "Formal Theories of Knowledge in AI and Robotics". In *New Generation Computing*, Vol. 3, No. 4, (special issue on Knowledge Representation), Ohmsha, Ltd., Tokyo, Japan (1985).

[Rosenschein and Kaelbling, 1986] Rosenschein, Stanley J. and Leslie P. Kaelbling. "The Synthesis of Digital Machines with Provable Epistemic Properties," *Proceedings of Workshop on Theoretical Aspects of Reasoning About Knowledge*, Monterey, California (1986).

[Kaelbling, 1988] Kaelbling, Leslie P. "Goals as Parallel Program Specifications." *Proceedings of the Seventh National Conference on Artificial Intelligence*, Morgan Kaufmann, St. Paul, Minnesota (August 1988).

[Kaelbling, 1987] Kaelbling, Leslie P. "Rex: A Symbolic Lanugage for the Design and Parallel Implementation of Embedded Systems." *Proceedings of the AIAA Conference on Computers in Aerospace*, Wakefield, Massachusetts (1987).

[Moszkowski, 1983] Moszkowski, Ben. Reasoning about Digital Circuits. Ph.D. Dissertation, Stanford University, Stanford, CA (1983).

[Goad, 1986] Goad, Chris. "Fast 3D Model-Based Vision." In *From Pixels to Predicates: Recent Advances in Computational and Robotic Vision*, Alex P. Pentland (ed.), Ablex Publishing Corporation, Norwood New Jersey (1986).

[Halpern and Moses, 1985] Halpern, Joseph Y. and Yoram Moses. "A guide to the modal logic of knowledge and belief." *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, California (1985).

[Doyle, 1979] Doyle, Jon. "A Truth Maintenance System." *Artificial Intelligence*, Vol. 12, No. 3 (1979).

[de Kleer, 1986] de Kleer, Johann. "An Assumption-Based Truth Maintenance System." *Artificial Intelligence*, Vol. 28, No. 1 (1986).

Exact Solution in Linear Time of Networks of Constraints Using Perfect Relaxation

Francesca Rossi*
MCC
3500 West Balcones Center Drive
Austin, Texas 78759
rossi@mcc.com

Ugo Montanari
Dipartimento di informatica
Università di Pisa
Corso Italia 40
56100 Pisa, Italy

Abstract

Relaxation algorithms of polynomial complexity are often used to explicitate constraint satisfaction problems as much as possible before the application of a backtracking solution algorithm. In order to better understand this technique, we propose a relaxation algorithm scheme, which can be instantiated to any real relaxation algorithm, and we study its properties and its complexity. Also, we identify the very convenient class of perfect relaxation algorithms, which have a complexity linear in the number of constraints of the problem to which they are applied and which explicitate the problem to the point of solving it. Finally, we characterise all the classes of constraint satisfaction problems for which it is easy to find a perfect relaxation algorithm by mapping them to the family of all the languages of context-free graph production systems.

1 Introduction

Networks of constraints are a simple knowledge representation method, useful for describing those problems whose solution is required to simultaneously satisfy several constraints. They are very important in AI because, for instance, problems like description of physical systems, scene interpretation and specification of software can often be naturally expressed as networks of constraints.

They are described here as labelled graphs, where nodes represent variables to be assigned, while arcs (i.e. hyperarcs connected to one, two or more variables) are constraints to be satisfied by the adjacent variables; constraints connected to k variables are labelled by relations specifying the acceptable k-tuples of values of the variables. A solution of a network of constraints is defined as an assignment of values to some (say k) variables which can be extended to the other (n-k) variables in order to obtain an assignment to all the variables (n) of the network satisfying all the constraints. These k selected variables are emphasised in the same definition of the network with an arc (called connection arc) connecting them. A network of constraints is solved if the label of its connection arc coincides with its solution. Note that this new notion of network of constraints, where a particular arc is selected, is not restrictive. Moreover, it is convenient for defining networks generated through arc-rewriting productions.

Unfortunately, both the problem of finding one solution and all the solutions of a network of constraints are NP-hard (both in our definition and in the usual one). This means that solution algorithms for general networks of constraints will have exponential worst case behaviours. But some solution algorithms can have acceptable performances in the average case.

For example, the generate and test algorithm will always be exponential, but the backtrack algorithm, which is the standard solution algorithm for networks of constraints, is linear in the best case and can be acceptable on the average. More precisely, the more the constraints of the network to be solved are explicit (they reflect the global constraint induced by the whole network), the more the backtrack algorithm works well. For this reason, many relaxation algorithms have been developed: given a network, they make some local transformations on it, and return a new more explicit network.

A relaxation algorithm can be abstractly defined as the application of some relaxation rules in some order, until no more changes can be done (we have the stable network). Each relaxation rule computes the solution of a subnetwork and possibly modifies, accordingly to this solution, a constraint of the network, making it more restrictive.

In this paper, we develop a convenient theory for representing both networks with k-ary constraints and general relaxation rules. Also, we define a relaxation algorithm scheme and we study its properties. In this way every particular relaxation algorithm can be seen as an instance of the scheme and can thus inherit its properties.

*Supported by a grant from the Italian National Research Council.

It can be easily shown that both the arc-consistency ([Mackworth, 1977]) and the path-consistency ([Montanari, 1974]) algorithm can be naturally described using our formalism, and the same holds also for some other relaxation algorithms that can be found in [Freuder, 1978, Freuder, 1982, Seidel, 1981].

Moreover, we show that the stable network obtained at the end of a relaxation algorithm depends only on the set of rules and not on their order of application (i.e. the relaxation rules, if applied an infinite number of times, satisfy the Church-Rosser property).

The set of rules used by a relaxation algorithm is called adequate if the stable network is solved, and perfect if it is adequate and the stable network can be achieved applying the rules only once, in an order called a perfect strategy. Perfect strategies are thus very convenient, provided that the rules are few, and small.

Also, we characterise particular classes of networks having relaxation rules of bounded complexity and where the number of rules needed by any network in the class is smaller or equal than the number of arcs in the network. These classes of networks are very interesting because, for every network in a class, it is possible to find a perfect relaxation algorithm which has linear complexity, even if the linearity coefficient may vary from class to class. The characterisation of such classes involves the definition of graph production systems (a kind of context-free graph grammar), and shows how each class can be mapped into the set of graph generated by one of such systems. Also, the perfect strategy for a network corresponds to the derivation (in the grapf production system) of such network. Each network in the characterised classes has a tree-like structure due to its generation through context-free production. More precisely, it can be seen as a tree where each node can be a (even cyclic) graph. Our result can be thus seen as a generalisation of a previous one ([Dechter and Pearl, 1988, Mackworth and Freuder, 1985]) about the existence of a linear algorithm for the solution of tree networks (networks that are exactly a tree). In fact here we can solve with linear complexity, by using a perfect relaxation algorithm, not only tree networks but also the much bigger family of all the classes of tree-like networks. It is also similar to the work described in [Shenoy and Shafer, 1988], where the perfect relaxation corresponds to the "computation of marginals". Also, a perfect relaxation algorithm can be seen as a reformulation, of a bottom-up algorithm, proposed in [Montanari and Rossi, 1988a], about the computation of the exact solution of hierarchical networks.

In section 2 we define networks of constraints in terms of connection graphs, in section 3 we define relaxation algorithms and we give some of their fundamental properties, section 4 treats perfect relaxation algorithms and defines graph production systems. Finally, section 5 concludes the paper summarising its results and arguing a possible relationship of these results with logic programming and metaprogramming.

2 Networks of Constraints

In this section we introduce connection hypergraphs and networks of constraints. For a more detailed description of networks of constraints and their properties see [Montanari, 1974, Montanari and Rossi, 1988b].

In the following of this paper, we will heavily deal with hypergraphs and hyperarcs, which are a natural generalisation of graphs and arcs respectively, where arcs may connect more than two nodes. Thus, to be simpler, from now on we will use the word graph to denote both graphs in their normal definition and hypergraphs, and the word arc to denote both arcs and hyperarcs.

Definition 1 *(Connection Graph)*
A connection graph $< a, G >$ *with* $G =< N, A, c >$ *consists of:*

- *a set of nodes* N;
- *a set of arcs* A; $A = \bigcup_k A_k$ *is a ranked set, i.e.* $h \in A_k$ *implies that* h *is an arc connected to* k *nodes;* $a \in A$ *and is called the connection arc;*
- *a connection function* $c : \bigcup_k (A_k \rightarrow N_k)$ *where* $c(h) =< x_1, \ldots, x_k >$ *is the tuple of nodes connected to* h.

If $c(a) =< x_1, \ldots, x_k >$, $A - a = \{a_1, \ldots, a_m\}$ *and* $c(a_i) =< x_{i1}, \ldots, x_{ik(i)} >$, *for* $i = 1, \ldots, m$, *then we will represent the* k*-connection graph* $a \leftarrow G$ *by writing* $a(x_1, \ldots, x_k) \leftarrow a_1(x_{11}, \ldots, x_{1k(1)}), \ldots, a_m(x_{m1}, \ldots, x_{mk(m)})$. ■

Definition 2 *(subgraphs)*
Given two connection graphs $a \leftarrow G$ *and* $b \leftarrow F$, $b \leftarrow F$ *is a subgraph of* $a \leftarrow G$ *if:*

- $A_F \subseteq A_G$ *(remember that* $b \in A_F$*);*
- $N_F \subseteq N_G$;
- $c_F = c_G$ *on* A_F. ■

Definition 3 *(networks of constraints)*
A network of constraints $C = a \leftarrow G \mid l$ *is a connection graph* $a \leftarrow G$, *where nodes are called variables and arcs are called constraints, plus a labelling function* $l : \bigcup_k (A_k \rightarrow \wp(U^k))$, *where* U *is a finite set of values for the variables of* C. ■

Definition 4 *(solution of a network of constraints)*
Given a network of constraints $C = a \leftarrow G \mid l$ *where* $G =< N, A, c >$ *and* $n =\mid N \mid$, *let us consider any ordering of the variables of* N, *say* $< x_1, \ldots, x_n >$. *Moreover, given an* n*-tuple* $< v_1, \ldots, v_n >= v$ *of values of* U, *let us set* $v_{|<x_{i1}, \ldots, x_{im}>} =< v_{i1}, \ldots, v_{im} >$. *Then, we define the solution of network* C *as* $Sol(C) = \{< v_1, \ldots, v_n >_{|c(a)}$ *such that, for all* $b \in A$, $< v_1, \ldots, v_n >_{|c(b)} \in l(b)\}$. ■

In words, the solution of a network of constraints is the set of all the assignments of the variables connected by the connection arc such that every such assignment can be extended to an assignment of all the variables in N which satisfies all the constraints in A.

An obvious, exaustive search algorithm to obtain the solution $Sol(C)$ of a network of constraints $C = a \leftarrow G \mid l$ is as follows.

Algorithm ES (exaustive search)

- Initialise $Sol(C)$ to the empty set;
- For every possible assignment v of values to all the variables in N:
  - If, for every constraint b in A, $v_{|c(b)} \in l(b)$, then add $v_{|c(a)}$ to $Sol(C)$.

The worst case time complexity of this algorithm can be estimated of the order of $(\mid U \mid)^{\mid N \mid} \mid A \mid\mid N \mid$.

Definition 5 *(solved networks of constraints)*
A network of constraints $C = a \leftarrow G \mid l$ is solved iff the label of its connection arc is equal to its solution, i.e. $l(a) = Sol(C)$.■

Using algorithm ES, an obvious way of solving a network of constraints $C = a \leftarrow G \mid l$ is obtaining $Sol(C)$ with this algorithm and then setting $l(a)$ to $Sol(C)$.

In fact, let us introduce the notation $l[a/b]$, where $l : A \rightarrow B$ is a total function, $a \in B$ and $b \in A$, for indicating the new function l' such that $l'(b) = a$, and $l'(c) = l(c)$ for all $c \in A$, $c \neq b$.

Now, the network $C_{sol} = a \leftarrow G \mid l[Sol(C)/a]$ is a solved network, and $Sol(C_{sol}) = Sol(C)$.

3 Relaxation methods

In this section we define relaxation algorithms as sequences of local changes to the labelling function of a given network. We also give the description of a general relaxation algorithm and we investigate some of its properties, while characterising two particular and interesting subclasses of relaxation algorithms.

Definition 6 *(relaxation rules)*
Given a network of constraints $C = a \leftarrow G \mid l$, a relaxation rule is any subgraph $b \leftarrow F$ of $a \leftarrow G$. Applying a relaxation rule $b \leftarrow F$ to $C = a \leftarrow G \mid l$ produces a new network of constraints C', which is: $C' = a \leftarrow G \mid l[Sol(b \leftarrow F \mid l)/b]$.■

Theorem 1 *(relaxation rules return equivalent networks)*
Given a network of constraints $C = a \leftarrow G \mid l$ and a relaxation rule $b \leftarrow F$, we have:
$Sol(a \leftarrow G \mid l) = Sol(a \leftarrow G \mid l[Sol(b \leftarrow F \mid l)/b])$.■

Definition 7 *(stable network)*
Given a set R of relaxation rules, a stable network w.r.t. R is a network $C = a \leftarrow G \mid l$ such that, for all r in R, r applied to C returns C.■

Lemma 1 *(stability and relaxation rules)*
Given a network of constraints $C = a \leftarrow G \mid l$ and a relaxation rule $b \leftarrow F$, the network $C' = a \leftarrow G \mid l[Sol(b \leftarrow F \mid l)/b]$ is stable w.r.t. the singleton set $R = \{b \leftarrow F\}$.■

Definition 8 *(strategies)*
Given a set R of relaxation rules, a strategy for R is a string $S \in R^* \cup R^\infty$. An infinite strategy S is fair if each rule of R occurs in S infinite times.■

Definition 9 *(relaxation algorithms)*
Given a network of constraints $C = a \leftarrow G \mid l$, a set R of relaxation rules, and a strategy S for R, a relaxation algorithm is an algorithm which applies to C the rules appearing in S until

- *a stable network (w.r.t. R) is obtained (in this case, this network will be called $C' = closure(C, S)$), or*
- *S terminates (in this case, the network obtained is more explicit than C but may be not stable w.r.t. R).*■

A Pascal-like description of a relaxation algorithm which receives as input a network of constraints C, a set $R = \{r_1, \ldots, r_n\}$ of relaxation rules, and a strategy $S = \{s_1, s_2, \ldots\}$ for R, is here given. Note that the marks are used for recording the rules of R which have to be applied (because they are adjacent to some rule which has changed some constraint of the network). The set of all the rules adjacent to a given rule is defined as $adj(b \leftarrow F) = \{b' \leftarrow F' \text{ in } R \text{ such that } b \in A_{F'}\}$.

```
Relaxation Algorithm RA(C, R, S):
begin
C' := C;
"mark all the rules in R";
i := 1;
while i ≤| S | do
begin
if "s_i is marked" then
  begin
  "unmark s_i";
  if C' = d ← H | l and s_i = b_i ← F_i then
    begin
    C_old := C';
    C' := d ← H | l[Sol(b ← F | l)/b];
    end;
  if C' ≠ C_old then "mark all the rules in adj(s_i)";
                else begin
                     if "all rules of R unmarked"
                       then return C';
                     end;
  end;
i := i + 1;
end;
return C';
end.
```

Theorem 2 *(RA returns an equivalent network)*
Given a network of constraints C, a set R of relaxation rules, and a strategy S for R, the relaxation algorithm $RA(C, R, S)$ always returns a network C' such that $Sol(C) = Sol(C')$.■

Moreover, note that, if S is finite, i.e. $S = s_1, \ldots, s_m$, the relaxation algorithm always terminates, either in less than or equal to m steps, returning a network stable w.r.t. R, or in exactly m steps, returning a network more explicit than the given one but not stable w.r.t. R.

If, on the contrary, S is infinite but fair, the following theorem holds.

Theorem 3 *(termination of a relaxation algorithm)*
Given a network of constraints C, a set R of relaxation rules, and a infinite fair strategy S for R, the relaxation algorithm $RA(C, R, S)$ terminates.■

Theorem 4 *(time complexity of RA)*
The worst case time complexity of algorithm RA is $O(\mid U \mid\mid rmax \mid (\mid R \mid + (\Sigma_{i=1,\ldots,\mid R\mid} \mid adj(s_i) \mid\mid U \mid^{\mid c(b_i)\mid}))) + dmax(\mid R \mid + (\Sigma_{i=1,\ldots,\mid R\mid} \mid adj(s_i) \mid\mid U \mid^{\mid c(b_i)\mid})))$, where $\mid rmax \mid = max_{j=1,\ldots,\mid R\mid} \mid r_j \mid$, and $\mid r_j \mid$ is the number of variables in rule r_j.■

The strategy plays a very important role in obtaining a stable network. In fact, if we have a finite strategy we may not obtain a stable network. On the contrary, if S is infinite, the obtained network not only is equivalent to C and stable w.r.t. R, but the following theorem shows that it does not depend on S, i.e., given C and R, there exists a unique network C' equivalent to C and stable w.r.t. R. Thus different infinite and fair strategies all lead us to the same network.

Theorem 5 *(the closure does not depend on S)*
Given a network of constraints C and a set R of relaxation rules, let us consider two infinite and fair strategies S' and S'' for R. Then, $closure(C, S') = closure(C, S'')$. Thus the closure of C depends only on R and we will write $closure(C, R)$.■

It can be easily shown that both the arc-consistency and the path-consistency algorithm are particular instances of RA. A simple transformation of algorithm RA, when applied to infinite fair strategies, will lead to a new relaxation algorithm $RA'(C, R)$, in which the strategy is built during the execution, as usual in the classical definition of relaxation algorithms. For RA', it can be shown that the worst case time complexity formula reduces to $O(\mid U \mid\mid rmax \mid (\mid R \mid + (\Sigma_{i=1,\ldots,\mid R\mid} \mid adj(s_i) \mid\mid U \mid\mid c(b_i) \mid)))$, which coincides, in every particular instance corresponding to a real relaxation algorithm, with the already known time complexity formula for that algorithm.

Thus, RA' is more efficient than RA, but it is equivalent to RA only when RA is applied to infinite fair strategies. In the following, we will be concerned mainly with finite strategies, thus we will consider RA as well.

Definition 10 *(adequate sets of rules)*
Given a network of constraints C and a set R of relaxation rules, R is adequate for C iff $C' = closure(C, R)$ is solved.■

In general, given a network C, both arc-consistency and path-consistency rules may be not adeguate for C.

Definition 11 *(perfect sets of rules)*
Given a network of constraints C, a set R of relaxation rules, and a strategy S for R which is a total ordering of R, R is perfect for C iff $RA(C, R, S)$ returns $C' = closure(C, S)$ and C' is solved. S is called a perfect strategy for R. RA in this case is called a perfect relaxation algorithm.■

Theorem 6 *(RA perfect is linear in $\mid R \mid$)*
Given a network of constraints C, a set R of relaxation rules, and a perfect strategy S for R, $RA(C, R, S)$ has a worst case time complexity $O(\Sigma_{i=1,\ldots,\mid R\mid} \mid U \mid^{\mid r_i\mid})$.■

4 Perfect relaxation problems and graph production systems

In order to have a perfect relaxation algorithm we have to be able to obtain a perfect strategy first. In this section we show how to do it in the case of networks that can be generated by a sequence of replacements of an arc by a subnetwork.

Definition 12 *(replacement)*
Given two graphs $a \leftarrow G$ and $a' \leftarrow G'$, and an arc b such that $b \in A$, $rank(b) = rank(a')$ and $b \neq a$, the replacement of b with $a' \leftarrow G'$ in $a \leftarrow G$ is the new graph $a \leftarrow H = a \leftarrow G[a' \leftarrow G'/b]$ where:

- *$N_H = (N \bigcup N')_{\mid E}$ where $c(a') E c(b)$;*
- *$A_H = (A \bigcup A')_{\mid L}$ where $a' L b$;*
- *c_H is the union of c and c'.*■

The notation $Q_{\mid E}$ in general denotes the set Q quotient the equivalent relation E. In particular, in our context this means that N_H contains all the nodes in N plus all the nodes in N', with the exception that the nodes in $c(a')$ and those in $c(b)$ are merged. The same is for A_H.

Lemma 2 *(replacement and solution)*
Given a network of constraints $C = a \leftarrow G[c \leftarrow H/b] \mid l$, we have: $Sol(a \leftarrow G[c \leftarrow H/b] \mid l) = Sol(a \leftarrow G \mid l[Sol(c \leftarrow H \mid l)/b])$.■

That is, replacement and solution satisfy a commutative property, in the sense that we can first replace and then solve or, equivalently, first solve and then replace. Obviously, the former alternative is more expensive, because we have to solve the entire network C, while in the latter alternative we have to solve two networks (C' and C'') of smaller size.

Definition 13 *(perfect relaxation problems)*
A perfect relaxation problem $a \leftarrow G \mid l$. S is a network of constraints $C = a \leftarrow G \mid l$ plus a perfect strategy S for a set R of relaxation rules for C.■

Definition 14 *(perfect uniform relaxation problem)*
$a \leftarrow G$. S is a perfect uniform relaxation problem if $a \leftarrow G \mid l$. S is a perfect relaxation problem for any labelling function l.■

Lemma 3 *(a simple perfect uniform relaxation problem)*
$a \leftarrow G$. $a \leftarrow G$ is a perfect uniform relaxation problem.■

Theorem 7 *(obtaining a new perfect uniform relaxation problem)*
$a \leftarrow G$. S perfect uniform relaxation problem implies $a \leftarrow G[c \leftarrow H/b]$. $(c \leftarrow H)S$ perfect uniform relaxation problem.■

Corollary 1 *(obtaining a perfect strategy)*
Given a network C in the structured form $C = a_1 \leftarrow G_1[a_2 \leftarrow G_2/b_1] \ldots [a_n \ldots G_n/b_{n-1}] \mid l$, then $S = (a_n \leftarrow G_n)(a_{n-1} \leftarrow G_{n-1}) \ldots (a_1 \leftarrow G_1)$ is a perfect strategy for R.■

Once a perfect strategy S for a given network C is obtained, the corresponding perfect relaxation problem $C.S$ can be solved by applying the relaxation algorithm RA, with the strategy S, to the network C. The returned network C' is equivalent to C, stable with respect to the rules in S and solved. Thus it is enough to take the label of the connection arc of C' to have the solution of the given network C.

To give a complexity result for our algorithm, when applied to perfect strategies, we define the notion of graph production systems. They are similar to context-free (hyper)graph grammars (for a survey and some related results on graph grammars, see [Ehrig, 1978, Habel and Kreowski, 1987, Montanari and Rossi, 1988a]).

Definition 15 *(graph production systems)*
A graph production system is a finite set P of connection graphs, called productions.■

Definition 16 *(language of a graph production system)*
Given a graph production system P, its language $L(P)$ is the (possibly infinite) set of all connection graphs of the form $a_1 \leftarrow G_1[a_2 \leftarrow G_2/b_1] \ldots [a_n \leftarrow G_n/b_{n-1}]$ where $a_i \ldots G_i$, for all $i = 1, \ldots, n$, is a production in P.■

Note that there are some classes of networks which cannot be included in the language of any graph production system. For example, the class of rectangular lattices cannot (see [Martelli and Montanari, 1972]).

Let us suppose to deal with graph production systems in which there are no empty productions, i.e. productions which do not add any arc to the given graph. This restriction is necessary to avoid an infinite number of steps to produce a finite graph, and thus to have a linear relationship between the number of steps used to generate a graph and the number of arcs in this graph. Let us call this kind of graph production system a Greibach graph production system. Then, the following result holds.

Theorem 8 *(RA is linear in the number of arcs)*
Given a Greibach graph production system P and a class of connection graphs GG such that $GG \subseteq L(P)$, algorithm RA, applied on each network $C = G \mid l$, where $G = a_1 \leftarrow G_1[a_2 \leftarrow G_2/b_1] \ldots [a_n \leftarrow G_n/b_{n-1}]$ in GG with the corresponding perfect strategy $(a_n \leftarrow G_n)(a_{n-1} \leftarrow G_{n-1}) \ldots (a_1 \leftarrow G_1)$, has a worst case time complexity linear with the number of arcs of G.■

5 Conclusions

In this paper we have introduced a general formalism which can be used for describing any relaxation algorithm as the application of some relaxation rules in a given sequence.

Moreover, we have defined perfect relaxation algorithms as relaxation algorithms which not only return a more explicit network, but also exactly solve the given network of constraints.

Finally, we have shown that perfect relaxation algorithms are very efficient when applied to particular classes of networks. They are all those classes of networks which are included in the language of some graph production system (a kind of context-free hypergraph grammar). Each network in a class can be solved by a perfect relaxation algorithm whose relaxation rules are directly derivated by the generation of the network, of bounded complexity and in number less or equal than the number of arcs of the network. Due to these properties of the rules, the algorithm returns the exact solution of each network with a time complexity which is linear with its size. Naturally, if the productions used for generating a network were given not in a sequential form, but as a tree, our algorithm migth have a parallel implementation and so a time complexity logarithmic with the size of the given network.

Besides that, note that the basic notation $a \leftarrow G$ used in this paper to represent connection graphs, networks of constraints and productions reminds logic programming. In fact, $a \leftarrow G$ can be seen as a Horn clause whose semantics is $Sol(a \leftarrow G \mid l)$ (if we add the definition of l as an extensional database). For this reason, an implementation of our relaxation algorithm scheme RA has been easily developed in Prolog using metaprogramming techniques (see [Rossi and Montanari, 1988]).

References

[Ehrig, 1978] Claus, Ehrig, and Rozenberg, editors. *Proc. International Workshop on Graph Grammars and their Application to Computer Science. Lecture notes in Computer Science*, Springer Verlag, Bad Honnef, 1978.

[Dechter and Pearl, 1988] R. Dechter and J. Pearl. Network-based heuristic for constraint-satisfaction problems. *Journal of Artificial Intelligence*, 34:1–38, 1988.

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

[Freuder, 1978] E.C. Freuder. Synthesising constraint expressions. *Communications of the ACM*, 21(11):958–966, November 1987.

[Habel and Kreowski, 1987] A. Habel and H.J. Kreowski. Some structural aspects of hypergraph languages generated by hyperedge replacement. In *Proc. STACS*, Springer Verlag, 1987.

[Mackworth, 1977] A.K. Mackworth. Consistency in networks of relations. *Journal of Artificial Intelligence*, 8(1):99–118, 1977.

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

[Martelli and Montanari, 1972] A. Martelli and U. Montanari. Nonserial dynamic programming: on the optimal strategy of variable elimination for the rectangular lattice. *Journal of Mathematical Analysis and Application*, 40:226–242, 1972.

[Montanari, 1974] U. Montanari. Networks of constraints: fundamental properties and application to picture processing. *Information Science*, 7:95–132, 1974.

[Montanari and Rossi, 1988a] U. Montanari and F. Rossi. An efficient algorithm for the solution of hierarchical networks of constraints. In Ehrig, editor, *Proc. Int. Workshop on Graph Grammars and Their Application to Computer Science*, Springer Verlag, 1988.

[Montanari and Rossi, 1988b] U. Montanari and F. Rossi. Fundamental properties of networks of constraints: a new formulation. In Kanal and Kumar, eds., *Search and artificial intelligence*, Springer Verlag, 1988.

[Rossi and Montanari, 1988] F. Rossi and U. Montanari. Relaxation in networks of constraints as higher order logic programming. In *Proc. Third Italian Conference on Logic Programming (GULP)*, Rome, May 1988.

[Seidel, 1981] R. Seidel. A new method for solving constraint satisfaction problems. In *Proc. IJCAI*, pages 338–342, 1981.

[Shenoy and Shafer, 1988] P.P. Shenoy and G.R. Shafer. *Constraint propagation.* Working paper 208, University of Kansas, School of Business, Lawrence, KS 66045-2003, November 1988.

Principles of Metareasoning

Stuart Russell* and **Eric Wefald**
Computer Science Division
University of California, Berkeley, CA 94720

Abstract

In this paper we outline a general approach to the study of formalized metareasoning, not in the sense of explicating the semantics of explicitly specified meta-level control policies, but in the sense of providing a basis for selecting and justifying computational actions. This research contributes to a developing attack on the problem of resource-bounded rationality, by providing a means for analysing and generating optimal computational strategies. Because reasoning about a computation without doing it necessarily involves uncertainty as to its outcome, probability and decision theory will be our main tools. We develop a general formula for the utility of computations, this utility being derived directly from the ability of computations to affect an agent's external actions. We address some philosophical difficulties that arise in specifying this formula, and some practical difficulties in the application of the formula to particular problem-solving systems.

1 Introduction

> Blot out vain pomp; check impulse; quench appetite; keep reason under its own control.
>
> *Marcus Aurelius Antoninus*

The study of resource-bounded intelligent systems promises to be a major area of research in AI in the near future, with the potential for drastic revision of our understanding of learning, inference and representation. One reason for this is practical: few people believe that classical, normative models can scale up. A second, and more fundamental, reason is that existing formal models, by neglecting the fact of limited resources for computation, fail to provide an adequate *theoretical* basis on which to build a science of artificial intelligence. The 'logicist' approach to AI emphasizes the ability to reach correct conclusions from correct premises. The 'rational agent' approach, derived from philosophical and economic notions of rational behaviour, emphasizes maximal achievement of goals via decisions to act. When resource bounds come into play, direct implementation of either approach results in suboptimal performance. Instead, what we want is an optimal design for a *limited* rational agent (LRA). A view of artificial intelligence as a *constrained* optimization problem may therefore be more profitable.[1]

Our project is the design of robust software architectures for goal-driven, resource-limited intelligent systems (also known as *ralphs*[2]). Our approach has been to address the design problem with the finitude of resources as a *starting point*, rather than trying to lop corners off the deductive model. A major tool of this research is a normative meta-level theory for the value of computations, since this allows an agent to allocate scarce computational resources optimally; alternatively, it allows the AI researcher to show the optimality of a non-optimizing algorithm for decision-making, and to design such algorithms constructively. Progress on developing and applying such a theory forms the main subject matter of this paper.

We begin in section 2 by defining the notion of real-time problem-solving, and discuss the various approaches that have been taken to the problem of boundedness in this context. Section 3 introduces the idea of *rational* meta-reasoning, wherein computations are treated as actions to be selected among on the basis of their utilities. In turn, these utilities are derived from the expected effects of the computations, chief among which are the consumption of time and/or space and the possible revision of the agent's intended actions in the real world. A formal model is set up for the meta-level decision problem, and, in section 4, we discuss the various situations in which exact and approximate solutions can be found for the util-

*This research was carried out with support from the AT&T Foundation and the Computer Science Division of the University of California, Berkeley. The second author is supported by a Shell Foundation Doctoral Fellowship. Mike Braverman did the figures.

[1] This view is developed in greater depth in [Horvitz and Russell, forthcoming].

[2] Ralphs inhabit the RALPH (Rational Agents with Limited Performance Hardware) project at Berkeley. The capability for self-adaptation to improve performance is a second focus of the project.

ity equations, as well as some possible approaches to the significant theoretical tasks that remain unsolved. Section 5 reports briefly on the results of our implementation efforts, for single-agent and game-playing search. We conclude with a discussion of future research directions.

2 Approaches to bounded rationality

Two conditions conspire to create what has been called the 'finitary predicament' [Cherniak, 1986]: first, real agents have only finite computational power; second, they don't have all the time in the world. The formal characterization of a 'real-time' problem situation is the following: the utility of a given action performed by the system exhibits a significant dependence on the time at which it is carried out, such that complete solutions to the decision problem are usually infeasible. Typically, the utility of an action will be a decreasing function of time. Since the time at which an action can be carried out depends on the amount of deliberation required to choose the action to be performed, there is often a tradeoff between the *intrinsic utility* of the action chosen and the *time cost* of the deliberation (see section 4.3 below). As AI problems are scaled up towards reality, virtually *all* problem situations will become 'real-time'. One aim of this paper is to develop a methodology for constructing real-time algorithms that can be used as the 'building blocks' for more complex systems.

Work in *real-time AI* has traditionally focused on delivering AI capabilities in applications demanding high performance and negligible response times. As a result, designers typically choose a fixed level of desired output quality, and design their systems to achieve that level within a fixed time cost. In a recent survey [Laffey et al., 1988], real-time systems are defined in terms of "the system's ability to guarantee a response after a fixed time has elapsed, where the fixed time is provided as part of the problem statement." Given this view of real-time, the authors are led to the conclusion that "research which focuses on speeding up a version of the algorithm that can guarantee a response time should be given high priority." As Dean [1988] has pointed out, this approach leads to a research strategy based on compile-time optimization and variants of job scheduling. It seems unlikely that such an approach will generalize, particularly to cases of widely varying time pressures and problem complexities. In addition, the 'deadline' model of time pressures is overly restrictive, since in reality there is almost always a continuous increase in the cost of time. Laffey *et al.* [1988] survey a large number of application programs for real-time AI, and note, somewhat despairingly, that "Currently, *ad hoc* techniques are used for making a system produce a response within a specified time interval."

A longer tradition of considering the effects of boundedness on decision-making exists in economics and the decision sciences, where human characteristics must sometimes be considered. Researchers in decision analysis, especially Howard [1966], have studied the problem of the *value of information*. His ideas have been applied for example to assess the benefits of expensive medical diagnostic procedures [Horvitz et al., 1988]. Simon [1982] made clear the distinction between systems that compute the rational thing to do (procedural rationality), and systems that simply do the rational thing (substantive rationality). Procedurally rational systems seem to suffer from a good deal of overhead, both in terms of time and extra cognitive machinery. A currently popular notion is that all this deliberation is a waste of time — why don't we just build agents that "do the right thing" [Brooks, 1986, Agre and Chapman, 1987]? Substantive rationality, however, does not come for free; in complex environments it can only be the result of previous *compiled deliberation*, on the part of the designer or the agent itself.

A premise of this research is that flexible, autonomous systems in complex environments require the ability to reason explicitly about the appropriate resources to allocate to computation at any point, and about which computations will be most effective. This premise is shared by several other researchers in AI, such as Doyle's 'rational psychology' project [Doyle, 1988]. In medical decision-making, rational control of probability calculation has been studied by Horvitz [1988] and by Heckerman and Jimison [1987]. Dean [1988] discusses optimal allocation of resources among processes in various resource-bounded scenarios. Fehling and Breese [1988] have applied a decision-theoretic approach to the control of information-gathering actions in a simple robot scenario. Agogino [1989] has investigated the use of decision-theoretic modelling of computation in the control of mechanical systems.

All meta-level systems share a common methodology for implementation: the base-level problem solver operates via the explicit formulation and solution of meta-level problems. In a *uniform* meta-level architecture, meta-level problems are formulated using the same language as the base-level problems, and solved using the same mechanism. This produces systems capable of great flexibility and high-order reasoning, but introduces the possibility of infinite regress. Regress is particularly problematic in the present context, since the metareasoning done to control problem-solving has costs itself, and therefore needs to be controlled. Regress has been mentioned by many researchers concerned with bounded rationality [Doyle, 1988, Horvitz and Russell, forthcoming, Laird et al., 1987, Fehling and Breese, 1988] but so far only Lipman [1989] has claimed any progress on the problem. Clearly we must back off from insisting on optimal control of *all* reasoning; some actions, whether computational or external, will have to be taken without being the *immediate* results of deliberation. For this reason, among others, two important topics in the RALPH project are inductive learning of meta-level policies and compilation of reasoning. Meta-level

learning is discussed briefly in section 4.5.1; on compilation of decision-making see [Russell, forthcoming]. Here we will not worry about the cost of metareasoning itself; in practice, we have been able to reduce it to an insignificant level.

3 Rational metareasoning

The construction of a system capable of rational metareasoning rests on two basic principles:

1. Computations are to be treated as actions, and are therefore to be selected among on the basis of their expected utilities.
2. The utility of a computation is derived from its expected effects, consisting of:
   (a) The passage of time (and associated changes in the external environment).
   (b) The possible revision of the agent's intended actions in the real world.

The ability of a computation to cause the agent to take a different course of action, that has been revealed by the computation to be superior to the agent's original intention, is the fundamental source of positive utility for computations. It is important to emphasize the obvious fact that the choice of which computation to make, and whether to continue computing, must be made in the absence of knowledge of the outcome of the computation (else the computation would be pointless). Therefore it will only be possible at best to perform optimally on average, by computing an *expected value* of the computation.[3]

3.1 Notation

- A_i: one of a set of possible external actions available to the agent in the current state.
- $[X]$: the result of taking action X in the current state, where the action can be internal (a computational action) or external.
- $[X, W_j]$: the result of taking action X in world state W_j.
- $U(W)$: a real-valued utility function on world situations W (reference to a particular agent is implicit).
- S_j: one of a set of possible computational actions available to the agent.
- **S**: a sequence of computational actions. We will use **T** to refer to a potential future sequence of computational actions, particularly one ended by an external action.

[3]Note that we do not make the same assumption about base level actions. That is, we assume that the outcomes of individual actions are known with certainty, as in game playing or puzzle-solving domains, although the utility of those outcomes may be uncertain. The case of uncertainty about the outcomes of base level actions will require separate treatment.

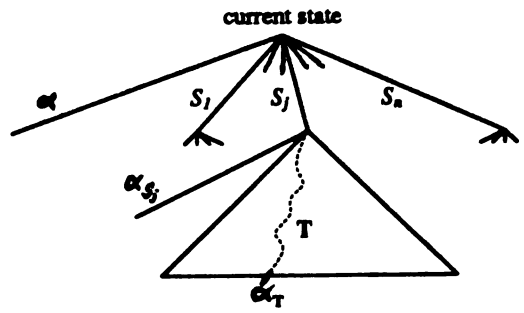


Figure 1: The metalevel decision situation

- $\mathbf{S}.S_j$: the sequence of actions consisting of sequence **S** followed by action S_j.
- $\hat{Q}^{\mathbf{S}}$: the agent's *estimate* of a quantity Q, calculated in the state resulting from a computation **S**.
- α: the agent's current default 'intention'; typically, the external action considered so far to have the highest utility.
- $\alpha_{\mathbf{T}}$: the external action recommended by a computation **T**.
- $\beta_1, \beta_2, \ldots$: the external actions currently ranked second-best, third-best etc.

3.2 The metalevel decision model

In order for the concept of resource allocation to make sense, the system in question must have a choice of computations available to it, each of which can return a decision or affect the ultimate decision made. The computations may vary along a number of dimensions; for our purposes, the amount of time used and the quality of solution returned will be most significant. Figure 1 illustrates the choice situation in which the agent finds itself. The agent's immediate choice is among the available computations steps $S_1 \ldots S_k$ and the current 'best' external action α, where typically[4]

$$\alpha = \text{argmax}_A\{\hat{U}^{\mathbf{S}}([A])\}.$$

Here we use $\hat{U}$, the agent's utility estimate, rather than U, the actual utility, since we want α to be the action that the agent *will* take, rather than the action that the agent *ought to* take. It is important to recall that, at this stage, we are considering the meta-level choice from the viewpoint of an external observer, without suggesting that the agent itself must explicitly set up and solve the decision problem for every computation. In particular, we do not assume that the agent has access to a rapidly-computable, exact utility function U.

[4]Initially, before any deliberation has been done, $\hat{U}$ will presumably be a constant. For definiteness, we will assume that ties are broken somehow, perhaps by random choice.

If the agent takes a computation step, then thereafter any sequence of steps may be taken, including the empty sequence, followed by an external action $\alpha_{\mathbf{T}}$, where $\mathbf{T}$ is the complete sequence of computation steps undertaken before acting. The computation steps can have any grain size, ranging from a single machine cycle to a multi-week economic simulation.

According to decision theory, an optimal action is one which maximizes the agent's expected utility, given by

$$E[U([A_i])] = \sum_j P(W_j)U([A_i, W_j]). \quad (1)$$

where $P(W_j)$ is the probability that the agent is currently in state W_j. The utility of a computation step S_j is defined, just as in ordinary decision theory, in terms of the resulting state $[S_j]$. Note that a computation directly affects the system's internal state, and only indirectly the external world (except by consuming time), whereas a utility function usually refers only to aspects of the total situation that are external to the agent (such as winning a game of chess) or to aspects of internal state such as happiness that are not generally accessible to direct computational manipulation.

3.3 The value of computation

We define the *net value* of a computational action to be the resulting increase in utility:

$$V(S_j) = U([S_j]) - U([\alpha]) \quad (2)$$

A major distinction that needs to be made in specifying $U([S_j])$ is between *partial* and *complete* computations. A partial computation is one that does not result in a commitment to external action; whereas a complete computation does.

If S_j is a complete computation, then the utility of S_j is just the utility of the action α_{S_j} chosen as a result of the computation, given that the action is carried out after S_j is completed. That is, $U([S_j]) = U([\alpha_{S_j}, [S_j]])$. Hence,

$$V(S_j) = U([\alpha_{S_j}, [S_j]]) - U([\alpha]) \quad (3)$$

In the general (partial) case, the computational action will bring about changes in the internal state of the agent that will affect the value of possible further computational actions. In this case, we want to assess the utility of the internal state in terms of its effect on the agent's ultimate choice of action. Hence the utility of the internal state is the *expected* utility of the base-level action which the agent will ultimately take, given that it is in that internal state. This expectation is defined by summing over all possible ways of completing the deliberation from the given internal state. That is, letting $\mathbf{T}$ range over all possible complete computation sequences following S_j, $\alpha_{\mathbf{T}}$ represent the action chosen by computation sequence $\mathbf{T}$, we have

$$U([S_j]) = \sum_{\mathbf{T}} P(\mathbf{T})U([\alpha_{\mathbf{T}}, [S_j.\mathbf{T}]]) \quad (4)$$

where $P(\mathbf{T})$ is the probability that the agent will perform the computation sequence $\mathbf{T}$.

If the agent has a perfectly rational metalevel, then the computation sequence selected will be the one maximizing $U([\alpha_{\mathbf{T}}, [\mathbf{T}]])$, and this sequence will have probability 1 in the above equation.

3.4 Ideal and approximate control

Given a means for calculating the value V of a computation rapidly and exactly, the *ideal control algorithm* is as follows:

1. Keep performing the most valuable of the available computations until none have positive value V.
2. Commit to the action that is preferred according to the internal state resulting from step 1.

This intuitive definition of the value of a computation, and the algorithm accompanying it, were proposed independently by Good [1968]. Obviously, the computation of the values V cannot be instantaneous; in fact, as we describe below, it can be arbitrarily hard. However, in many cases calculation is sufficiently fast to result in benefits over other control methods. In other cases, the formulation of a formally correct estimation of the value of computation results in a tractable meta-level after compilation, or forms the basis for well-motivated approximations.

For the purposes of control of real agents, the equations given above need considerable modification. Usually, the agent only has an exact utility available for a small subsets of possible states. For example, in chess only won, lost and drawn games are accurately evaluated. However, often the agent does have available an *estimate* $\hat{U}$ of the utility of a state. In the AI literature, such estimation functions are often called evaluation functions. We can view the process of decision-making as *revising the estimated utility function*, by learning more about the consequences of actions and their utilities. This is the general picture of deliberation about how to act on which we will most closely focus our attention. For instance, in a game-playing program, a computation might involve a further ply of search, followed by back-up of new minimax values for the top-level move choices. These new backed-up values would give the agent's new estimated-utility function, based on its latest computation. Note, however, that we make no general assumptions about how the estimated utility function is arrived at.

4 Calculating the value of computation

In this section, we will spell out transformations on the equations given in the previous section, which render them more useful to a less-than-omniscient designer or to a limited rational agent that is explicitly reasoning about its own problem-solving.

Our first job is to replace the function U by the function $\hat{U}$, since we are assuming that the former is

unknowable, and the agent is able to calculate only in terms of the latter. This will require some care, because the function $\hat{U}$ depends on the stage of the computation. For reasons that we discuss below, we choose to replace U in equation 4 by utility estimates for actions made in the state in which the external action is taken. Thus, if we assume that the agent will take whatever action appears best at the time it decides to act, then equation 4 becomes

$$\hat{U}^{\mathbf{S}.s_j}([S_j]) = \sum_{\mathbf{T}} \hat{P}^{\mathbf{S}.s_j}(\mathbf{T}) \max_i \hat{U}^{\mathbf{S}.s_j.\mathbf{T}}([A_i, [S_j.\mathbf{T}]]) \quad (5)$$

Then the estimated net value of computation S_j, from equation 2, is given by

$$\hat{V}(S_j) = \hat{U}^{\mathbf{S}.s_j}([S_j]) - \hat{U}^{\mathbf{S}.s_j}([\alpha]) \quad (6)$$

Of course, before the computation S_j is performed, $\hat{V}(S_j)$ is a random variable. Although the agent can't know ahead of time what the exact value of $\hat{V}(S_j)$ will be, given sufficient statistical knowledge of the distribution of $\hat{V}$ for similar actions in past situations, the agent can take its *expectation*,

$$E[\hat{V}(S_j)] = E[\hat{U}^{\mathbf{S}.s_j}([S_j]) - \hat{U}^{\mathbf{S}.s_j}([\alpha])] \quad (7)$$

4.1 Analysis for complete computations

Suppose we know that the agent will act after the computation step S_j in question (if it does not act immediately); i.e., suppose it is only choosing between complete computations. Then the utility of S_j will be equal to the utility of the action α_{S_j} believed by the agent to be optimal after S_j has been carried out, given that the action is carried out *after* S_j is completed. That is,

$$\hat{U}^{\mathbf{S}.s_j}([S_j]) = \hat{U}^{\mathbf{S}.s_j}([\alpha_{S_j}, [S_j]]) \quad (8)$$

In this case, therefore, the net value of S_j is given by

$$\hat{V}(S_j) = \hat{U}^{\mathbf{S}.s_j}([\alpha_{S_j}, [S_j]])) - \hat{U}^{\mathbf{S}.s_j}([\alpha]) \quad (9)$$

In sections 4.2-4.4, we show how this formula can be simplified by employing various assumptions about the nature of the domain and the search methods employed. The general case of partial computations is much more complex. In section 4.5, we discuss some possible approaches to the general case, although we do not have a complete solution for the general case at this time.

4.1.1 Discussion

Note that in the above equations we use the later estimate $\hat{U}^{\mathbf{S}.s_j}$ to evaluate both α_{S_j} and α, the new and the old best moves. The reason for this is most obvious if we consider the case where $\alpha_{S_j} = \alpha$, but $\hat{U}^{\mathbf{S}.s_j}([\alpha_{S_j}]) > \hat{U}^{\mathbf{S}}([\alpha])$; i.e., S_j simply revises upward the estimated utility of α but does not alter the choice of move. Then the net value of S_j should depend only on the passage of time spent in deliberation, since the real intrinsic utility is of course constant. This would not be the case if we used the later utility estimate for the new best move and the current utility estimate for the current best move.[5]

This points to deep issues which arise from the fact that the agents we are dealing with have only limited rationality. For instance, Howard's Information Value Theory [Howard, 1966] uses a formula which, in the above context, amounts to defining the *expected* net value of S_j as

$$E(\hat{V}(S_j)) = E(\hat{U}^{\mathbf{S}.s_j}([\alpha_{S_j}, [S_j]]) - \hat{U}^{\mathbf{S}}([\alpha])) \quad (10)$$

This will only be equal to our formula, given in equation 9, *provided* $E(\hat{U}^{\mathbf{S}.s_j}([\alpha])) = E(\hat{U}^{\mathbf{S}}([\alpha]))$. This *coherence condition* will hold true of a perfectly rational agent, since any expected increase in its expected utility estimate due to further deliberation should already be reflected in the current estimate.

However, for an agent with only limited rationality, for instance an agent who relies on a fixed, easily computed evaluation function for his utility estimate, it may not be safe to assume that his estimate is rational in this sense. However imperfect the base-level problem solver is, we must assume as a premise that the real utility of carrying out a given action at a given time won't be changed just by thinking about it. This principle is even more important in *inductive meta-level learning*, wherein the agent evaluates computational actions *post hoc* in order to learn statistical distributions and a predictive function for the value of computation (see section 4.5.1). Here, we do not want to use the formula whose expectation is taken in 10, since this will lead to erroneous evaluations. Rather, in such evaluations it is important to use the same estimation function for the values of both the new and the old best actions. For these reasons, we use equation 9 to estimate the net value of a computation. Note that equation 9 also gives the *agent's* best estimate of the value of the computation, *at the time the computation is completed.*

4.2 The variation of estimated utility with computation

The formula for the value of a computation given in equation 9 can be evaluated provided we have statistical data concerning the effects of similar computations carried out in the past.

Briefly, suppose we can characterize S_j as belonging to a given *class* of computations, i.e. suppose S_j is a particular *type* of computation, such as an additional ply of search in an iterative-deepening algo-

[5] Of course, the desired effect would be obtained if we used the current estimates for both moves. But then every computation would have non-positive utility, since by definition α has a higher current utility estimate than the other moves.

rithm, being considered in a particular type of situation, where the situation is characterized by some pre-determined set of features. Then, using a very crude approach, we can characterize the distribution of the random variable $\hat{V}(S_j)$ to almost any desired standard of approximation by computing *post hoc* the net increase $\hat{U}([\alpha_{S_j}, [S_j]]) - \hat{U}([\alpha])$ for a large sample of computations in similar situations drawn from the same class. Of course, computing the actual increase for a large sample of similar situations will be much more expensive than simply carrying out the computation S_j. However, if a large amount of "design time" is available, in which such statistical sampling can be done without time pressure, with the resulting distributions stored in a parametrized form for later use, then the cost of applying formula 9 to estimate the expected net value of computation S_j can be orders of magnitude cheaper than carrying out S_j itself. In that case, the expected value calculation will be well worth doing, since it allows one to select among computations, to prune pointless branches, and to terminate deliberation in such a way as to maximize the overall utility of the agent.

This empirical approach can clearly be refined. The refinements require knowledge of the base-level decision-making methods of the agent. Essentially, the principle is this:

1. Typically, a computation under consideration is known to affect only *certain components* of the agent's internal structure; for example, running a query about proposition p through a belief network will affect the probability that the agent assigns to p.

2. Changes in those components affect the agent's choice of action in known ways; for example, if the base-level is decision-theoretic, then a change in the probability assigned to some action outcome would affect the expected utility for that action, and hence the choice of action, in the manner prescribed by equation 1.

The empirical component of evaluating computations arises only in the first of these two stages; the more specific we can be about the structure of the base level, the easier it will be to focus on a well-defined, homogeneous population of computation episodes and the more accurate our value estimates will become.

Let $p'_{ij}(u)$ be the normalized prior probability density function for the value of $\hat{U}^{\mathbf{S}.S_j}([A_i, [S_j]])$, the future utility estimate after the computation S_j has been carried out. Let $p_{\alpha j}$ be the density function for $\hat{U}^{\mathbf{S}.S_j}([\alpha])$. S_j can in general affect the utility estimates for any of the actions A_i. Thus, let $\mathbf{u} = < u_1, \ldots, u_n >$, and let $p_j(\mathbf{u})$ be the joint distribution for the probability that the actions A_1 through A_n get new utility estimates u_1 through u_n respectively. Then, taking the expectation of the right-hand side of equation 9, we have

$$E(\hat{V}(S_j)) = \int_{\mathbf{u}} \max(\mathbf{u}) p_j(\mathbf{u}) d\mathbf{u} - \int_{-\infty}^{\infty} u\, p_{\alpha j}(u) du \quad (11)$$

4.3 Time and its Cost

Thus far we have captured the real-time nature of the environment by explicitly including the situation in which an action is taken in the argument to the utility function. We believe that any form of time constraint can be expressed in this way. However, the inclusion of this dependence significantly complicates the analysis. Under certain assumptions, it is possible to capture the dependence in a separate notion of the *cost of time*, so that the consideration of the quality of an action can be done separately from considerations of time pressure.

For many estimated utility functions $\hat{U}$, we can define a related function, the estimated *intrinsic utility*, denoted by $\hat{U}_I$, along with a cost function C, that expresses the difference between total and intrinsic utility:[6]

$$\hat{U}([A_i, [S_j]]) = \hat{U}_I([A_i]) - C(A_i, S_j) \quad (12)$$

where $\hat{U}_I([A_i]) = \hat{U}([A_i])$.[7]

Of course there will always exist a function C which will satisfy equation 12, if only trivially. In order for $\hat{U}_I$ to qualify as an intrinsic utility, it must satisfy a further constraint. Namely, once the state $[S_j]$ has become the new current state, the agent's optimal action at that time should always be the one with highest intrinsic utility, independently of the cost C. A sufficient condition for this is that the cost be independent of the action:[8]

$$\hat{U}([A_i, [S_j]]) = \hat{U}_I([A_i]) - C(S_j) \quad (13)$$

Moreover, in the cases we are considering the change in actual utility of an action that occurs during some computation S_j will depend only on $\|S_j\|$, the length (in elapsed time) of S_j, and on the course of events in the outside world during that time. Since, by definition, computations alter only internal state, S_j will not

[6] Often in AI applications, we begin with an intrinsic utility function (such as a static evaluation function), and C is then defined to yield an accurate estimate of the true utility of an action under various time pressures. Only $\hat{U}$ is in fact independently definable from empirical observations.

[7] We define intrinsic utility with reference to the current situation only for convenience; in fact, utility is well-defined only up to an arbitrary linear transformation.

[8] This is approximately true in many AI domains such as game-playing or path-planning in a fixed or slowly changing environment. It will not be true in domains such as hunting or war, where different possible actions will gain or lose value at very different rates over time. Even in chess this condition can sometimes fail, since as time is used up more complex positions become less valuable.

affect that course of events, so we can let C, and thus $\hat{U}$, depend on the length of the computation rather than the computation itself. Thus the cost function, which we will in this case call TC or "time cost", gives the loss in utility to the agent due to the temporal delay in performing any given action:

$$\hat{U}([A_i, ||S_j||]) = \hat{U}_I([A_i]) - TC(||S_j||) \quad (14)$$

If such a function TC exists, then we can separate out the cost and benefit of a computation. We can therefore rewrite equation 9 as follows:

$$\begin{aligned} \hat{V}([S_j]) &= \hat{U}^{\mathbf{S}.S_j}([\alpha_{S_j}, [S_j]]) - \hat{U}^{\mathbf{S}.S_j}([\alpha]) \\ &= \hat{U}_I^{\mathbf{S}.S_j}([\alpha_{S_j}]) - \hat{U}_I^{\mathbf{S}.S_j}([\alpha]) - TC(||S_j||) \\ &= \Delta(S_j) - TC(||S_j||) \end{aligned} \quad (15)$$

where Δ denotes the *benefit* of the computation.

4.4 Subtree Independence

Given a utility estimation function which is separable into cost and benefit components in the above way, in many cases we can be more precise about the effects of a computation and thus obtain a further simplification of the above equation. Often, the agent design is such that a given computation affects the utility estimate of only one action. This is the case of *subtree independence*. In many domains, such as standard game-playing programs using minimax as a back-up method, this assumption will be straightforwardly true, if we consider individual node expansions as single computational steps.[9] Thus, subtree independence holds if and only if

$$\hat{U}_I^{\mathbf{S}.S_j}([A_i]) = \hat{U}_I^{\mathbf{S}}([A_i]) \quad (16)$$

for all actions A_i except at most the one action whose utility estimate is affected.

In this case, examining equation 11, we see that there are essentially two distinct cases in which a computation can have positive benefit, by changing the agent's intention. Either further computation about some currently non-preferred move β_i causes its utility estimate to be raised above that of α, or computation on α causes its utility estimate to be lowered below that of β_1, the current second-best move. Let us call two such computations S_j and S_k respectively (see figure 2).

Suppose we are considering the computation S_j, which affects only the estimated utility of action β_i. The search action will only change our preferred move if $\hat{U}_I^{\mathbf{S}.S_j}([\beta_i]) > \hat{U}_I^{\mathbf{S}.S_j}([\alpha])$, or equivalently, given

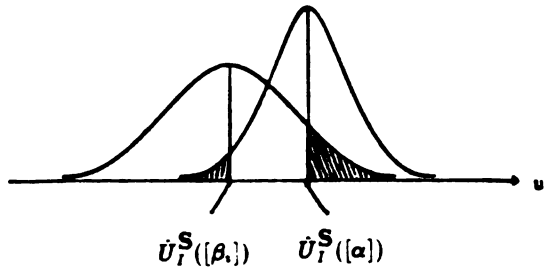


Figure 2: Effects of computation on action preference

equation 16, only if $\hat{U}_I^{\mathbf{S}.S_j}([\beta_i]) > \hat{U}_I^{\mathbf{S}}([\alpha])$. (the shaded region to the right of $\hat{U}_I^{\mathbf{S}}([\alpha])$ in figure 2). If this happens, we expect to be better off by an amount $\hat{U}_I^{\mathbf{S}.S_j}([\beta_i]) - \hat{U}_I^{\mathbf{S}}([\alpha])$— otherwise, there is no gain since our move preference remains unchanged. Thus in this case

$$E[\hat{V}(S_j)] = \int_{\hat{U}_I^{\mathbf{S}}([\alpha])}^{\infty} p_{ij}(x)(x - \hat{U}_I^{\mathbf{S}}([\alpha]))\mathrm{d}x \; - \; TC(S_j) \quad (17)$$

where $p_{ij}(x)$ is the probability density function for the random variable $\hat{U}_I^{\mathbf{S}.S_j}([\beta_i])$.

Similarly, if we perform a computation S_k that affects only the utility estimate of the current best action, our action preference is changed only if $\hat{U}_I^{\mathbf{S}.S_k}([\alpha])$, the new expected value of our current preferred move, is less than $\hat{U}_I^{\mathbf{S}}([\beta_1])$. (Although the new estimated utility of the new preferred action would be less than the current estimated utility of the current preferred action, the agent would still be *better off than it was*, since the computation will have revealed that α is a blunder.) Hence

$$E[\hat{V}(S_k)] = \int_{-\infty}^{\hat{U}_I^{\mathbf{S}}([\beta_1])} p_{\alpha k}(x)(\hat{U}^{\mathbf{S}}(\beta_1) - x)\mathrm{d}x - TC(S_k) \quad (18)$$

where $p_{\alpha k}(x)$ is the probability density function for $\hat{U}_I^{\mathbf{S}.S_k}([\alpha])$.

4.5 Partial computations

If we could always assume that the agent would necessarily take the action α_{S_j} after performing search action S_j, then calculating the value of computation would always be as simple as in the above paragraphs. However, in general this is not the case, since, as long as the agent does not arrive at complete certainty about its utility function — which we assume our agents almost never attain — in state $[S_j]$ the agent will still have a choice between taking action α_{S_j}, and continuing to deliberate (assume, for simplicity, that there is only one course of computational action open to the agent at each juncture). Thus the value of the computation S_j will be the value of having this choice. There are at least two ways to model the utility of

[9] However, if the search space is treated as a graph rather than a tree, as in some chess and go programs, then the analysis becomes slightly more complicated. In certain problem-solving systems the full analysis must be used. For example, if a computation involves refining a probability estimate used in an influence diagram, the new value may affect the utility of all the top-level actions.

being in the state of having a choice between actions, which we discuss below. We include this section for the sake of completeness, although the problem of evaluating partial computations raises very complex issues which we have not gotten to the bottom of.

4.5.1 An adaptive approach

If we assume that the agent will be able, when faced with such a choice, to choose the best option, then the utility of having a choice between two actions is just the maximum of the utilities of the two actions. This is a standard approach taken in decision analysis, for instance by Howard [1966]. Let S_k be the action of continuing to compute in state $[S_j]$. Then on this model, the utility of computational action S_j in the current state is given by

$$E[U([S_j])] = E[\max\{U([\alpha_{S_j}, [S_j]]), U([S_k, [S_j]])\}] \quad (19)$$

One difficulty with this formula is that it defines the value of a present computation in terms of the value of a possible future computation. Moreover, it is usually unreasonable for the agent to assume that it will necessarily choose the action with maximum utility in state $[S_j]$.

On the other hand, if we replace U by $\hat{U}$ in equation 19, we obtain the following:

$$E(\hat{U}([S_j])) = E(\max\{\hat{U}([\alpha_{S_j}, [S_j]]), \hat{U}([S_k, [S_j]])\}) \quad (20)$$

It might be argued that equation 20 is indeed a principle of rationality, in the sense that it imposes a constraint on any *extension* of $\hat{U}$ from base-level actions to computational actions. For it is true almost by definition that the agent will pick the action with maximum $\hat{U}$, and hence in a sense equation 20 says that the agent should value an action the same as it values the computation that recommended the action.[10]

In any case, equation 20 still cannot define a *unique* extension of $\hat{U}$ to computational actions because it is "ungrounded"; i.e. it does not specify how the recursion bottoms out. One possible way of overcoming the non-uniqueness involves specifying a *conservative extension* of $\hat{U}$ from base-level to computational actions. In fact, all we need is a method for arriving at a reasonable *post hoc* evaluation of a given computational action, in order to gain knowledge about the distribution of values by statistical sampling. Note that at the time we are collecting our sample data, we can have available to us the complete outcome of any given decision-making event. Thus, if we are willing to assume, solely for the purpose of evaluating sample computations, that our current agent's decisions between action and computation are correct, then we can arrive at justifiable numerical values by evaluating completed computations using equation 15, and backing up the values obtained to determine the values of partial computations. A more detailed description of this procedure is given in [Russell and Wefald, 1988].

Data derived in the above way will of course be error-prone, since the whole point of doing the analysis is that we think that the agent often makes incorrect judgments concerning when to stop and when to continue computing. But once we have done our sampling and equipped the agent with decision-theoretic search control knowledge, it will in a sense no longer be the same agent, since it will make different, and we hope better, choices in the same sorts of situations. But once an initial set of distributions is obtained, it would be a simple matter for the agent to revise those distributions incrementally as it gains more decision-making experience. In this way, the agent might adaptively converge on a state in which it would possess accurate knowledge of the value of its own computational procedures.

A number of technical problems need to be solved before the ideas of this section can be implemented. The most important one is that we will need to find easily-computable features of the search trees which will discriminate situations according to the distribution of the value of further search. We believe that in game-playing domains, one important such feature will almost certainly be the *conspiracy number of the search tree*, where this is defined here, by a slight alteration of McAllester's [1988] definition, as the minimal number of leaf nodes whose values must change in order to change the current best move choice. It is clear that as this number increases, the cost of achieving a given increase in the utility of the chosen move increases as well. In this way, we hope to make McAllester's valuable insights a part of a rigorous decision-theoretic approach to game-tree search control.

4.5.2 Probabilistic self-modelling

If we do not wish to assume that the agent will always be able to choose the best action in state $[S_j]$, we may assume instead that the agent has a certain probability of taking any given action. This probability may be directly related to the utility of the action; in an extreme case, if we assume that the probability is 1 for the action with highest utility, and 0 for other actions, we arrive at the model of equation 19. The better the agent's utility estimator $\hat{U}$ as an approximation to U, the closer will these probabilities come to this extreme case, but as long as $\hat{U}$ remains error-prone, the probabilities will lie in the open interval $(0, 1)$.

Let $p_j(\alpha_{S_j})$, $p_j(S_k)$ be respectively the probabilities that in state $[S_j]$ the agent will immediately take the new best action α_{S_j}, and that it will continue de-

[10] Against this it might be argued that if the agent *knows* that its estimated-utility function is error-prone, it needn't think that the right-hand side of equation 20 gives the true expected utility of the computational action S_j. We have found it very difficult to settle the sorts of philosophical questions raised by such considerations, and will not attempt to do so here. But we hope the dilemma raised here might convince the reader, as it has us, that the questions raised here are very deep.

liberation with computational action S_k. Then on this model the expected utility of the computational action S_j is given by

$$E[U([S_j])] = E[p_j(\alpha_{S_j})U([\alpha_{S_j},[S_j]]) + p_j(S_k)U([[S_k,[S_j]])] \quad (21)$$

Again, this formula gives the value of a current computational action in terms of the value of a future one. However, if we expand the action S_k recursively in a similar way to the expansion of S_j, and so on, we develop the familiar *decision tree* characteristic of such situations.

Moreover, if we expand all computational actions in this way, the right-hand side of equation 21 becomes the expectation of a sum of terms of the form $p_j q_j(\alpha_0) + p_k q_k(\alpha_1) + \ldots,$, where the α's are selected from the base-level actions A_i, and q_k represents the probability of reaching the kth node, that is, $\prod_{i=0}^{k-1}(1-p_i)$. This expression thus averages over all possible complete computations (see equation 4).

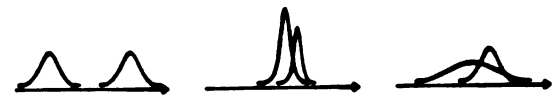
If we then combine terms corresponding to situations in which the same action is chosen, we get an expression of the form $p_1(A_1)U(A_1) + \cdots, + p_n(A_n)U(A_n)$, where now p_i represents the *probability that action A_i is eventually chosen* after completing the computation. Thus, after transforming the equation in this way, we can then drop the expectation sign on the right-hand side:

$$\hat{U}^{\mathbf{S}.S_j}([S_j]) = \sum_i \hat{\Pr}^{\mathbf{S}}(A_i)\hat{U}^{\mathbf{S}}([A_i]|A_i \textit{ chosen}) \quad (22)$$

where we use the conditional notation to denote informally that the fact of an action being chosen influences our estimate of its utility, and the fact that the action's utlity will depend on the time at which it is taken, and hence on the computation that ends in its being recommended. We believe it should be possible in practice to estimate the various probabilities and conditional expected utilities involved in this equation, although we have not yet attempted to implement the ideas of this section.

4.6 Qualitative behaviour

Before looking at specific applications, we can describe the *qualitative* behaviour of any algorithm based on our approach. Clearly, an agent will tend to forego further consideration of an action whenever its current estimated value and that of the best candidate are too far apart; in this case, it is unlikely that further computation will provide useful information, since the probability of changing action preference with any reasonable amount of extra computation is negligible. But deliberation may also be pointless if the current estimated utilities of two actions are too close together; in that case, it may be unlikely that further computation will reveal a *significant* difference between them. In an extreme case, the two actions may actually be symmetric, or nearly so, so that no amount of computation will differentiate significantly between them. This case has received scant attention in the literature, since many algorithm designers erroneously assume that the goal of deliberation is to identify the best action. Lastly, if there is considerable uncertainty as to the values of the available actions, and considerable overlap, further computation is recommended. We illustrate the three major situations graphically in Figure 3.



(a) terminate (b) terminate (c) continue

Figure 3: Three basic situations

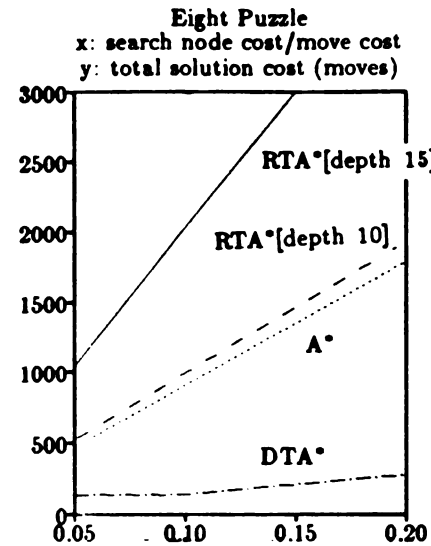
5 Applications

In this section we briefly describe how the formulæ above can be realized in a practical system. In our implementations to date, we have not attempted to evaluate accurately all partial computations, but rather in a number of domains we have obtained quite positive results by employing what we call the *single-step assumption*. This is the assumption that, for the purposes of evaluation, each possible computation step can be treated as a complete computation. Thus, we employ the formulæ of sections 4.2-4.4, which result in very tractable calculations. In order to apply equations 17 and 18 we need information about the distribution of possible effects of a computation on the system's value estimates for its available actions. This information will depend on the nature of the individual domain. Here space permits only the briefest summary of the practical results we have obtained in a few domains; for precise details of the analysis and implementation involved we must refer the interested reader to [Russell and Wefald, 1988].

In a search program, computation typically proceeds by expanding 'frontier nodes' of the partially-grown search tree. The value estimates for actions are calculated by backing up values from the new leaf nodes through their parents. The effect on the value estimate for an action is therefore composed of two aspects: the effect on the value of the leaf nodes that are expanded, and the transmission of this effect back through the tree. The first component depends on the nature of the node being expanded, and the nature of the expansion computation. Statistical information on the probability distribution can therefore be acquired by induction on a large sample of similar states using the same type of expansion computation. The second component is an analytic function of the state of the tree (in particular, the current value estimates for the nodes on the path to the root, and their children) that depends on the backing-up method used by the search program.

For single-agent search with an admissible heuristic, we have shown [Russell and Wefald, 1988] that only nodes in the subtree of the current best move should be expanded; derived a computable formula for the expected benefit for expanding a set of such nodes; shown that a best-first search (such as A*) is in fact the best policy given only the heuristic function; and derived an optimal stopping criterion for real-time search. Two common applications of this type of search are the familiar eight-puzzle, and path planning through an environment containing randomly-shaped polygonal obstacles. If we model the cost of computation by assuming a fixed cost-ratio between the cost of an edge in the search graph and the cost of the corresponding motion in the external world, we arrive at a figure for the total cost of a solution to such a problem, by summing the cost of the solution path taken and the cost of the search employed to find it. We have constructed an algorithm, called DTA* for "decision-theoretic A*", which attempts to minimize the expected total solution cost using the ideas discussed here. We have tested DTA*, for which the cost-ratio is a given parameter, against both A*, which always finds shortest paths regardless of search cost, and RTA* [Korf, 1987], which uses a limited search horizon. Typical results for the eight-puzzle and for a 20-polygon path-planning problem are shown in figure 4. In these applications, the cost of the meta-level computations performed by DTA* was insignificant.

For game-playing, we have derived a formula for the value of expanding a leaf node that can be computed with very little overhead, given some simplifying assumptions [Russell and Wefald, 1988]. In addition to the single-step assumption, mentioned above, we also employ what we have called the "meta-greedy assumption". This is the assumption that only individual node expansions need be considered as alternative computations (hence the resulting algorithms are greedy at the meta-level, or "meta-greedy"). These two assumptions lead to a fairly simple, easily computable formula for the expected value of a node expansion, which can be used to control and terminate search and has proven quite effective, up to a point. We call the resulting game-tree search algorithm MGSS*. In this case, however, the use of the single-step and meta-greedy assumptions together imposes a restriction on the effective search-depth of the algorithm. This is because as the search tree grows larger it is highly likely to reach a state such that no *single* node expansion can alter the best move choice; at that point, given our simplifying assumptions, the algorithm must conclude that no further search can be worthwhile.[11] We call this phenomenon the "meta-greedy barrier". In practice, for the game of Othello we have found that MGSS* outplays a standard alpha-beta algorithm searching to depth 3 by roughly 2 to



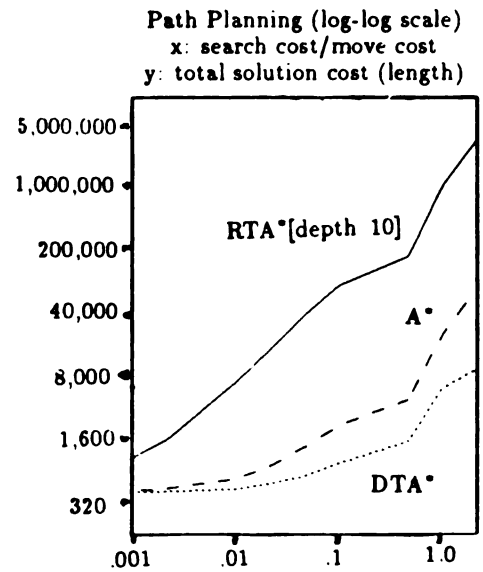


Figure 4: 8-puzzle and path-planning performance

[11] This is the point at which the "conspiracy number" of the tree, as we have re-defined this term above, goes from 1 to 2.

	MGSS*	α-β
Time	1095 sec.	1261 sec.
#nodes	190,077	279,697
#wins	22	10

Table 1: Summary of results for Othello

1, while only doing 2/3 as much search, in terms of number of nodes generated (results in terms of time used are slightly less favorable, due to the overhead of the meta-level computations). The results of a particular 32-game tournament, involving 16 random starting positions, with each side alternately playing black and white, are summarized in table 1.

Jeff Conroy has applied our approach to build a backgammon program, for which the preliminary outlook is quite favorable, especially since deep searches are not possible in backgammon due to the enormous branching factor. We are currently working to generalize our formulæ to overcome the restrictions mentioned, and hope to incorporate it into the next version of Berliner's HITECH system.

6 Further work

In addition to trying to relax the meta-greedy and single-step assumptions to obtain still better search control, we are currently pursuing three research directions to extend the range of applicability of the principles of metareasoning we have developed.

The first aims to improve the efficiency of real-time interleaving of computation and action by preserving and extending the partial search tree developed by previous deliberations. Our current implementations successfully alternate deliberation and action, but do not retain any information between deliberations.

The second project concerns improving the space-efficiency of selective-search algorithms. The selective search paradigm has come under criticism because of the need to keep a significant portion of the search tree in memory. A recursive, or problem-reduction algorithm can be implemented that still employs the metareasoning methods described above yet only keeps a small number of nodes in memory. The approach is based on *iterative expansion*, a generalization of iterative deepening. The algorithm is given a current state and a resource limit (say K nodes of search), and calls itself on the state's successors with resource allocations $k_1 \ldots k_n$. Depending on the results of the initial search, the algorithm can increase the resource allocation of some successor by a constant factor α. In this way the algorithm uses only linear space, and in the limit wastes a negligible amount of time in repeated subtree expansions. Further, the cost-benefit comparison of possible search steps would be done at the top level of each recursive call, rather than in a best-first fashion over the whole subtree.

The third, and most important, research project is to extend the analysis from evaluation search to other forms of decision-making computations, and to construct a general problem-solving architecture that employs 'normative' control over all its activities. The concept of *universal subgoaling* [Laird, 1984, Laird et al., 1987] is intended to capture the notion of a complete decision model, by making every aspect of the agent's deliberation recursively open to metareasoning. In the SOAR system [Laird et al., 1987], however, the basic deliberation mode is goal-directed search. We intend to construct a problem-solving architecture in which decision-theoretic deliberation and its various possible compilations [Russell, forthcoming] are the basic modes of computation, and in which metareasoning is carried out in the principled fashion outlined above, rather than through human-supplied meta-level productions.

One can also consider the possibility of applying these ideas to control search in theorem provers. However, in order to do this one needs something amounting to a 'current best move'. As currently implemented, theorem provers have no partial information about the success of the branch under consideration. The notions of 'guaranteed solution' and 'proof' must be replaced by *tentative/abstract/partial solution* and *justification*. Algorithms using defaults, abstraction hierarchies and island-driving strategies thus seem more amenable to meta-level control and therefore much more robust in the face of complexity.

7 Summary

We see computational resource limitations as a major influence on the design of optimal agents. This influence has been neglected in classical theories of normative behaviour, with the result that practical AI systems for non-trivial domains are constructed in an *ad hoc* fashion. A formal theory of the value of computation will play a central role in the design of optimal limited rational agents.

The basic insight behind normative control reasoning is that computations are actions. Choosing good actions involves reasoning about outcomes and utilities. The utility of a computational action *must be derived from its effect on the agent's ultimate choice of action in the real world.* The next problem is to assess this effect without actually performing the computation. When the base-level problem-solver operates using value estimates for the real-world actions, this can be done by using prior statistical knowledge of the distribution of the new value after the computation in question. The required distributions can be induced from direct observation of actual computations. Estimates of the value of computations can then be used to optimize a system's overall behaviour in real-time situations. We formalized the notion of real-time problem-solving in our framework, and identified time cost as a useful abstraction enabling significant simplifications in the theory. We have applied the theory to analyze both single-agent problem-solving and competitive game-playing, in each case yielding new algo-

rithms with improved performance. Performance systems for a variety of domains can be derived by making suitable approximations to the normative formula for meta-level control, and by proving associated simplifying theorems. We indicated some areas for further research in this vein. We also noted some qualitative insights into principles of resource allocation, pruning and termination.

Overall, a decision-theoretic, meta-level architecture, particularly one endowed with a varied set of forms of compiled knowledge, generates a rich space of possible agent configurations. But more importantly, the fact that all the components in such a system have a well-defined semantics, and the fact that they are connected by normative inferential links, means that an agent can make well-motivated changes in its own configuration as it searches the space of possible designs for an optimal state.

References

[Agre and Chapman, 1987] Agre, P. and Chapman, D. (1987) Pengo: An implementation of a theory of activity. In *Proc. 6th National Conference on Artificial Intelligence*, Seattle, WA: Morgan Kaufmann.

[Agogino, 1989] Agogino, A. M. (1989) Real-time reasoning about time constraints and model precision in complex, distributed mechanical systems. In *Proc. AAAI Symposium on AI and Limited Rationality*, Stanford, CA: AAAI.

[Brooks, 1986] Brooks, R. A. (1986) A robust, layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, **2**(1), 14-23.

[Cherniak, 1986] Cherniak, C. (1986) *Minimal Rationality*. Cambridge: MIT Press.

[Dean, 1988] Dean, T., and Boddy, M. (1988) An Analysis of Time-Dependent Planning. In *Proc. 7th National Conference on Artificial Intelligence*, Minneapolis, MN: Morgan Kaufmann, 49-54.

[Doyle, 1988] Doyle, J. (1988) *Artificial Intelligence and Rational Self-Government*. Technical report no. CMU-CS-88-124, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA.

[Fehling and Breese, 1988] Fehling, M. R., and Breese, J. S. (1988) A computational model for decision-theoretic control of probkem-solving under uncertainty. In *Proc. 4th Workshop on Uncertainty in AI*, Minneapolis, MN: AAAI.

[Good, 1968] Good, I. J. (1968) A five year plan for automatic chess. *Machine Intelligence*, **2**.

[Heckerman and Jimison, 1987] Heckerman, D., and Jimison, H. (1987) A perspective on confidence and its use in focusing attention during knowledge acquisition. In *Proc. Third Workshop on Uncertainty in AI*, Seattle, WA: AAAI, 123-131.

[Horvitz, 1988] Horvitz, E. J. (1988) Reasoning about beliefs and actions under computational resource constraints. In Uncertainty in Artificial Intelligence Vol. 3., (T. Levitt, J. Lemmer, and L. Kanal, eds.), Amsterdam: North Holland.

[Horvitz et al., 1988] Horvitz, E. J., Breese, J. S., and Henrion, M. (in press) Decision theory in expert systems and artificial intelligence. *Journal of Approximate Reasoning*, to appear.

[Horvitz and Russell, forthcoming] Horvitz, E. J., and Russell, S. J. (forthcoming) AI and limited rationality. In preparation.

[Howard, 1966] Howard, R. A. (1966) Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, **SSC-2**(1), 22-26.

[Korf, 1987] Korf, R. E. (1987) Real-time heuristic search: First results. In *Proc. 6th National Conference on Artificial Intelligence*, Seattle, WA: Morgan Kaufmann, 133-138.

[Laffey et al., 1988] Laffey, T. J., Cox, P. A., Schmidt, J. L., Kao, S. M., and Read, J. Y. (1988) Real-time knowledge-based systems. *AI Magazine*, **9**(1), 27-45.

[Laird, 1984] Laird, J. E. (1984) *Universal Subgoaling*. Doctoral dissertation, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA.

[Laird et al., 1987] Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987) SOAR: An architecture for general intelligence. *Artificial Intelligence* **33**, 1-64.

[Lipman, 1989] Lipman, B. (1989) How to decide how to decide how to ... Limited rationality in decisions and games. In *Proc. AAAI Symposium on AI and Limited Rationality*, Stanford, CA: AAAI.

[McAllester, 1988] McAllester, D.A. (1988) Conspiracy Numbers for Min-Max Search. *Artificial Intelligence*, **35**, 287-310.

[Russell and Wefald, 1988] Russell, S. J., and Wefald, E. H. (1988) *Decision-theoretic control of search: General theory and an application to game-playing*. Technical report UCB/CSD 88/435, Computer Science Division, University of California, Berkeley, CA.

[Russell, forthcoming] Russell, S. J. (forthcoming) Execution architectures and compilation. Paper submitted to IJCAI-89, Detroit, MI.

[Simon, 1982] Simon, H. A. (1982) *Models of Bounded Rationality, Volume 2*. Cambridge: MIT Press.

Combining logic and differential equations for describing real-world systems

Erik Sandewall*
Department of Computer and Information Science
Linköping University
Linköping, Sweden

Abstract

The paper shows how to combine non-monotonic temporal logic with differential calculus. A temporal logic is defined where time is real-valued and not discrete, and where real-valued, continuous parameters are used with their derivatives. Differential equations can therefore be used directly as axioms, and need not be transformed into confluences. This logic is used for characterizing common-sense physical systems where some parameters, or some of their derivatives, are occasionally discontinuous. Differential calculus then serves for characterizing the parameters during time-segments where they are continuous, and logic is used for characterizing the parameters around the discontinuity points. Models and preferential entailment is defined for this logic. For a simple scenario example (ball falling into shaft) it is shown what geometrical and physical axioms are needed, and how the axioms preferentially entail the desired common-sense conclusion.

1 Introduction and Approach

This paper describes how simple physical systems, of the kinds that we all understand and reason about by common sense, can be characterized and analyzed using a combination of differential equations and formal logic. The proposed method is an alternative to the contemporary standard approach in qualitative reasoning, which uses discrete value spaces[Hay79] and confluences[dKB85].

The method applies to *piecewise continuous* physical systems, i.e. systems whose possible histories over time may contain a number of significant time-points called *breakpoints*, and where all parameters[1] of the system are assumed to be continuous in the intervals between the breakpoints. At the breakpoints however the parameters may have discontinuities: the *left limit value* $a^l(t)$ and the *right limit value* $a^r(t)$ are defined for the parameter a in the breakpoint t, but need not be equal.

One example where breakpoints arise is when a held object is let go, so that its vertical acceleration (i.e. the second derivative of its vertical coordinate) changes instantly from 0 to -9.81 metric. Another example is when an amount of water is being heated and reaches boiling temperature, at which point the first derivative of its temperature switches from non-zero to zero, in a sufficiently simplified account.

Differential calculus is a fine instrument for characterizing the parameters in their continuous intervals between the breakpoints, but by its very nature the differential calculus does not deal well with discontinuities. The method described here is to use logic formulas (= wffs) in a suitable temporal logic for characterizing the behavior of the quantitative parameters around the breakpoints, and in particular for constraining the left and right limit values of parameters in breakpoints.

Furthermore a non-monotonic logic is used, since we need to state a default that the left limit value and right limit value are equal even in breakpoints if it is consistent with the axioms for them to be so. More precisely, we shall use preferential entailment[Sho88] under a preference relation which chronologically minimizes the instances of parameter discontinuity. The proposed preference criterium can be seen as a generalization of frame-problem persistence, from the classical view in A.I. of a discrete time axis and discrete properties, to our approach using real-valued time and real-valued, piecewise continuous parameters.

One can think of total histories for such a system as being constructed from parameter segments, each parameter segment being a continuous function over a closed finite interval on the real time axis. Each available parameter segment is characterized using differential equations. Non-monotonic logic, using appropriate axioms describing the object system and expressing

*This research was supported by the Swedish Board of Technical Development and the IT4 research program

[1] The set of parameters is defined so that the derivative of a parameter is also a parameter, recursively. Parameters are real-valued functions of time, corresponding to what is often called fluents in the A.I. literature.

physical laws, is used as the gluing agent which puts the parameter segments together.

2 What is wrong with the standard approach to qualitative reasoning

The method described here is proposed as an alternative to the present standard approach to qualitative reasoning. The standard approach has evolved through a sequence of significant papers[Hay79, dKB85, For81, Kui82], and has developed a consensus about its basic tenets. Before the new method is described in the next section, we shall briefly discuss the reasons for trying something different from the standard. - The abbreviation QR will be used for qualitative reasoning, and DC for differential calculus.

2.1 Need for a unified theory of temporal reasoning

"Qualitative reasoning", "temporal reasoning", and "knowledge based planning" are now three distinct areas in A.I. with very little technical overlap. However they are all concerned with phenomena over time, and it would be very useful to have a unified theory, presumably based on a temporal logic, which includes stringent accounts of both qualitative reasoning and planning. The research reported here is one step towards such a unified theory.

2.2 Conceptual economy of being consistent with other disciplines

The standard approach to qualitative reasoning involves the reconstruction, for discrete value spaces, of many concepts and results from DC: arithmetic for qualitative values, confluences, a discrete counterpart of the mean-value theorem, etc. This enterprise is by far not yet finished; it remains to map geometry, many remaining physical laws, and so on to the discrete-value domains. A priori it is much more elegant to use the original domains of real-valued time and real-valued parameters as domains for the logic, and to use the original differential equations as axioms.

2.3 Partial information must influence axioms, not interpretations

One of the common arguments for the standard approach to QR is that DC descriptions are over-specific since they assume availability of information we do not have. This is a faulty argument, simply since equality is not the only allowable relation between quantities: inequalities are also a part of the notation! For example the information that a rolling ball moves forward with positive velocity, under the influence of friction, air resistance, and other more or less random factors, can be just as well conveyed by

$$\partial x > 0$$

as by

$$\partial x = +$$

From the model theoretic point of view, the first formula allows us to use interpretations where parameters have real values, whereas the second formula forces us to use interpretations where the parameter values are taken from a discrete set. Of course the set of all models will have much greater cardinality if parameters have real values, but so what?

Scarcity of information about the object system must be reflected in a scarcity of information in the set of axioms describing that system, but need not be reflected in a scarcity of information in each particular interpretation for the axioms.

2.4 Differential calculus is declarative, and therefore multi-purpose

Another repeatedly used argument for the standard approach is that DC descriptions can only be used for two purposes, namely for finding their analytic solution, or as a basis for numerical calculation to find the solution in a specific instance. This argument is also incorrect: deduction is a perfectly appropriate use of quantitative descriptions, particularly when inequalities are involved. One obvious example is when it is known that

$$f(a) \leq c \leq f(b)$$
$$a \leq b$$

and that f is continuous in the interval between a and b, one can conclude

$$\exists x[a \leq x \leq b \wedge f(x) = c]$$

Another example is when it is known that

$$d(t_0) < 0$$

where d is a parameter and therefore a function of time, and that

$$\partial d \geq k > 0$$

where k is a constant and is therefore independent of time, to conclude that

$$\exists t_1[t_1 > t_0 \wedge d(t_1) = 0]$$

at least with appropriate assumptions regarding breakpoints and discontinuities.

2.5 Comparing continuous and discrete value spaces for applications without breakpoints

There are of course also systems which do not have breakpoints, or (more precisely) which one chooses to describe as fully continuous and not having breakpoints. For such systems, the standard approach using algebraic operations in discrete value spaces has to compete with logical reasoning about inequalities (and of course also equalities) in DC. It has never been

stated explicitly why one should prefer the former over the latter.

One possible reason might be notational elegance and conciseness. Discrete value spaces sometimes win in this respect, since one may write algebraic rules such as

$$+ * - = -$$

instead of the more clumsy logical formula

$$a > 0 \wedge b < 0 \rightarrow a * b < 0$$

The algebraic formulation is particularly advantageous for larger expressions which can be evaluated with discrete values for each sub-expression. This however assumes that operations such as + and * are always defined in the discrete value space, and in particular that the value of a composite expression is a function of the values of its components. That is not the case since for example the value of

$$+ - +$$

is undetermined in the discrete value space $\{+, 0, -\}$. The problem is aggravated when other landmark values than zero are admitted.

If discrete value spaces are preferred on the grounds of notational elegance and simplicity, then one would at least need to know that that advantage really holds in a systematic comparison, and that no significant loss of expressiveness occurs as a result. Neither of those issues has been studied (to my knowledge).

2.6 Inference capability

Yet another of the reasons commonly quoted for the standard approach to QR is that there are certain conclusions about the physical systems under consideration, which people can easily obtain by common sense, but which can not be obtained from the DC descriptions of such systems. Therefore, it is argued, one has to develop alternative and A.I.-style descriptions, based on conceptual representations of the types that people are assumed to use.

We have already argued that DC descriptions lend themselves to more kinds of symbolic manipulation than has often been assumed for the sake of the argument. However there is also a deeper issue: since the switch from a quantitative to a qualitative system description involves a *loss* of information, how can it possibly enable one to draw *more* conclusions than would otherwise have been possible? If something is being said in the qualitative formulation which can not be said in the quantitative one, then exactly what is that something?

The answer must be that additional deductive principles have been introduced in the QR representation using the qualitative domain, for example in the guise of constraint propagation mechanisms which are commonly used. The exact nature of the new mechanisms is however unclear, or at least has not usually been spelled out in clear logical terms in the QR literature.

Our position is that any additional deductive principle which is needed for common sense reasoning about physical or mechanical systems, should be developed, if at all possible, for use together with quantitative descriptions using DC, and not only for qualitative descriptions.

In particular, the results in the present paper show how frame- problem style persistence methods, can be adapted and generalized so as to also apply for continuous time and continuous parameters.

It is interesting to note that the use of discrete quantity spaces did not predominate as much in the early work on qualitative reasoning. The NEWTON system of de Kleer[dKl77], for example, stores mathematical equations in frames. One of the key points with NEWTON is that it contains multiple problem solving techniques, including both qualitative and quantitative methods, which can be applied to the conventional mathematical representation. It is in later papers that de Kleer and other authors introduce the method of "preprocessing" the problem statement into formulations using a discrete quantity space, which does away with the possibility to use problem-solving methods that use the DC representation.

2.7 What is really needed for doing qualitative reasoning

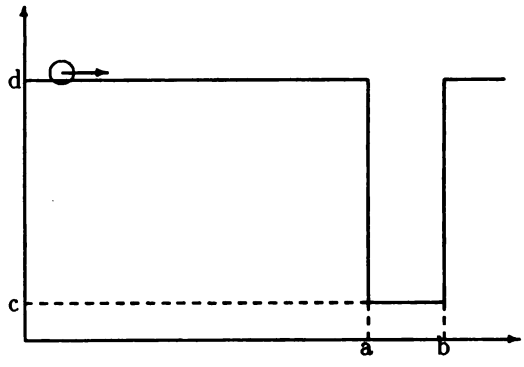
In summary, we claim that the standard approach to QR using discrete value spaces, confluences, etc. is a dead end. It originated from an incorrect analysis of whether one could, and why one could not obtain common sense conclusions from conventional descriptions of physical systems using differential calculus.

For fully continuous systems (without breakpoints i.e. all parameters are continuous at all times), we believe that reasoning about inequalities, equalities, and differential equations is sufficient for obtaining common-sense conclusions.

For object systems which are only piecewise continuous, something however has to be added, namely a descriptive machinery for characterizing - and constraining - the values of the parameters around the breakpoints. Switching from continuous to discrete value spaces is not an adequate response to that need, except if it were the only way to introduce logical constraints at all. In the present paper we show how logical constraints on breakpoint values can be directly combined with descriptions of the object in differential calculus.

3 Example scenario

Consider a simple, vertical two-dimensional mechanical system, such as the one illustrated in the following figure:



A ball is moving along a horizontal "plane" (actually a horizontal line) towards a shaft with vertical sides and a horizontal bottom. Idealized physical laws are assumed: the ball has zero size; there is no drag so the ball's horizontal velocity is constant until it reaches an obstacle, and when the ball reaches the pit its vertical acceleration changes instantly from zero to -9.81. The ball bounces perfectly against the walls of the pit, so that its vertical velocity does not change and its horizontal velocity reverses its sign but keeps the magnitude. Also the bottom of the shaft absorbs impact perfectly, so the ball stops its vertical movement without bounces when it reaches the bottom.

An obvious common sense conclusion from the scenario (scene description + "physical" laws) is that the ball will end up on the bottom of the shaft. We shall show below how that common sense conclusion is logically entailed by axioms expressed directly in real-valued calculus.

It may be objected that there are no discontinuities in actual physical systems. No actual ball starts falling instantly, nor does it make perfect bounces, or stop instantly on the bottom of shafts. This is true but beside the point, since the piecewise continuous description is a good approximation for many physical systems, and in particular can be expected to be good enough as a basis for reasoning.

In general, many interesting mechanical systems can be well characterized as having a number of "modes", "phases", or "states", where differential equations can be used to characterize the relationships between parameters within each of the modes with sufficiently good approximation. At the border between the states those approximations are not sufficient. The simplest possible way of characterizing the system around the shift of state is to assume that some of the parameters are discontinuous there. For common-sense reasoning, such simple approximations are of particular interest.

4 Interpretations and Formulas

We now proceed to the substance of the new method. In pursuing the approach that was outlined in the previous sections, the following three things need to be done first:

1. define the structure of interpretations in the logic
2. define the syntax of logic formulas (= wff)
3. define semantic entailment in this logic

We view the development of an inference system and inference procedure matching the semantic entailment as a secondary issue which can be pursued later on, once semantic entailment has been defined. We also consider it natural to define interpretations first and syntax afterwards.

4.1 Interpretations

An interpretation in our logic is a sixtuple

$$\langle M, U, S, R, Q, W \rangle$$

where M is a set of mode symbols, U is a set of parameter symbols closed under ∂, $S \subseteq \mathbf{R}$ is a "sparse" set of time-points (where $\mathbf{R}$ is the set of real numbers), namely the breakpoints, $R : (\mathbf{R} \times M) \hookrightarrow \{T, F\}$ assigns a truth-value to each mode at each point in time, $Q : (\mathbf{R} \times U) \hookrightarrow \mathbf{R}$ similarly assigns a value to each parameter at each point in time, and W is a mapping from temporal constant symbols to $\mathbf{R}$.

In addition interpretations must satisfy the following requirements:

1. $Q(t, u)$ as a function of t is continuous over every interval not containing a member of S,
2. if $t \notin S$ then $d/dt\ Q(t, u) = Q(t, \partial u)$,
3. $R(t, m)$ as a function of t is constant over every interval not containing a member of S.

It is in fact possible to get rid of M and R from the definition by encoding modes as parameters whose value is always either of 0 or 1, but only at the expense of understandability.

4.2 Syntax of logic formulas

We introduce the syntax for formulas by examples taken from the ball-and-shaft scenario above:

$$\Box \partial^2 x_b = 0$$

where $x_b \in U$ is a parameter or "fluent" representing the horizontal position of the ball, so that $Q(t, x_b)$ is the x-coordinate of the ball at time t. The set U was assumed to be closed under the syntactic operation of prefixing a ∂ symbol, so ∂x_b, $\partial^2 x_b$, etc. are also in U. The value of $Q(t, \partial^2 x_b)$ is the horizontal acceleration of the ball at time t if $t \notin S$, but is not constrained to equal the acceleration if $t \in S$. The operator $\Box$ means that the formula following it is true for all $t \in \mathbf{R}$, and

has the weakest possible priority so that it extends to the end of the formula or to the next closing right parenthesis.

$$\Box Wall(x_b, y_b) \rightarrow \partial x_b^l = -\partial x_b^r \wedge \partial y_b^l = \partial y_b^r$$

where $Wall(x, y)$ is true iff the point (x, y) in the vertical plane is a wall point, i.e. x is either a or b, and y is strictly between c and d. Also the l and r "exponents" are used to signify left and right limit value as follows:

$$Q(t, \partial x_b^l) = \lim_{\tau \rightarrow t-0} Q(\tau, \partial x_b)$$

is the horizontal velocity of the ball "just before" impact against the wall at time t. The whole formula expresses the idealized physical law that on impact against the wall, the ball reverses the sign of its horizontal velocity component instantly and retains its vertical velocity.

Notice that both the above axioms can be satisfied in the same interpretation. There is an apparent conflict because if the horizontal velocity has a discontinuity, as the second axiom implies, then the horizontal acceleration can not be always zero. However this is not a problem since $Q(t, \partial^2 x_b)$ is only required to equal the horizontal acceleration for those t not in S. If the interpretation is only constructed in such a way that the time of bounce t_0 is included in S, then $Q(t, \partial^2 x_b)$ may equal zero also for $t = t_0$ although $d/dt\ Q(t, \partial x_b)$ is not zero for $t = t_0$, and is in fact not even defined there.

$$\Box Supp(x_b, y_b) \rightarrow \partial y_b^r = 0$$

where $Supp(x, y)$ is true for those points (x, y) located on a horizontal supporting "surface". This axiom says that an object which is already on such a supporting surface stays there, and that an object which falls onto the surface stops without bouncing back up again. (However, as we shall see below this axiom has to be formulated somewhat differently in order to exclude all interpretations where the object "falls through" the surface).

It is straightforward to extrapolate the appropriate syntax for logic formulas from these examples, and to complete the set of axioms characterizing the application. However what we have described so far does not allow one to refer to "observations" i.e. statements about the value of a parameter at a specific point in time. The notation so far is only sufficient for characterizing physical laws (such as the rule relating the parameters involved in a perfect bounce) and for describing the physical structure of the scenario, for example the existence and location of the shaft (assumed to be held fixed over time). We shall refer to this information as the *permanent axioms* and write it as Π.

4.3 Observation axioms

In addition to the permanent axioms there is also a need for axioms describing the state of the system at specific points in time. We shall refer to such axioms as observation axioms.

An appropriate notation for expressing observations may be as

$$[t_1]\ x_b < a \wedge y_b = d$$

for saying that the ball is on the supporting surface and to the left of the shaft at time t_1, or

$$[t_1]\ \partial x_b > 0$$

for saying that the ball is moving towards the right at time t_1. Formulas such as these may be assigned a value by interpretations as defined above, since the W component of interpretations assigns a time-point as value to each time-constant such as t_1.

Actually observations have to be recorded as logical formulas of the form

$$[t_1]\ \partial x_b > 0 \wedge C(\partial x_b)$$

where C is a predicate saying that the parameter appearing as its argument, is continuous at the present time-point. We will soon come to the reasons why this is necessary.

If Π is a set of permanent axioms, and Γ is a set of observation axioms for *one* time-point t_1, then we should expect the model set for $\Pi \cup \Gamma$ to be the set of all "histories" i.e. temporal developments which are compatible with the observations in Γ. The types of reasoning which are commonly called prediction (temporal projection) and postdiction (temporal explanation) can therefore be understood as identifying formulas α which satisfy

$$\Pi \cup \Gamma \models \exists t_2 [t_1 < t_2 \wedge [t_2]\ \alpha]$$

and

$$\Pi \cup \Gamma \models \exists t_2 [t_1 > t_2 \wedge [t_2]\ \alpha]$$

respectively.

Also if Γ is a set of observations at *two* points in time, the model set and the entailed conclusions can be interpreted as an explanation of observations (given that the facts in Γ have actually been observed, conclude what must have happened in between). If the logic is extended so that actions are also admitted, then the conclusions from $\Pi \cup \Gamma$ in this case may alternatively be interpreted as a plan (or more precisely, as a disjunction of all available plans) for getting from the first "observed" state to the second "observed" state. In this case the second state is of course the goal statement for the plan.

The case where Γ is the empty set, finally, corresponds to the reasoning process called *envisioning* by de Kleer[dKl77]. In other words, the model set for Π alone should be the set of all possible histories in the world, as constrained by the laws of "physics" and physical structure, but not by any particular observations of the state of the system.

5 Preferred models and entailment

We have now defined the structure of interpretations, and (outlined) the syntax of logical formulae. We proceed to the third issue that was specified initially, namely defining the criteria for the model set, and therefore the entailment relation for the present type of logic.

The result is as follows. It is not adequate to use ordinary, monotonic entailment, where the set of models for the axioms is simply the set of those interpretations wherein all the axioms are true. The reason is that the axioms may specify under what circumstances a parameter must have a discontinuity, e.g. by specifying the relationship between its old and new value (like in the axiom for the perfect bounce above), but it is difficult or impossible to specify through separate axioms all the cases where a parameter should not have a discontinuity in those time points belonging to S. There is therefore a need for a non-monotonic principle saying that parameters are continuous at break-points whenever that is consistent with the axioms.

Non-monotonic inference can be characterized in a model theoretic framework using preference relations over interpretations, as has been demonstrated by Shoham[Sho88]. A preference relation $\ll$ over interpretations is defined as follows.

Definition. If $J = \langle M, U, S, R, Q, W\rangle$ is an interpretation, and t is a time point, then the *breakset* i.e. the set of discontinuities at time t in J is

$$breaks(J,t) = \{m \mid R^l(t,m) \neq R^r(t,m)\} \cup \{u \mid Q^l(t,u) \neq Q^r(t,u)\}$$

Clearly if t is not in S then the breakset of J at t is empty.

Definition. If $J = \langle M, U, S, R, Q, W\rangle$ and $J' = \langle M', U', S', R', Q', W'\rangle$ are interpretations, then we write $J \ll J'$ and say that J is preferred over J' iff there is some time-point t_0 such that

1) for all $t < t_0$, all $m \in M$ and all $u \in U$, $R(t,m) = R'(t,m)$ and $Q(t,u) = Q'(t,u)$

2) $breaks(J,t) \subset breaks(J',t)$.

Definition. If Π is a set of permanent axioms i.e. axioms not containing any explicit references to time-points, and if α is another logic formula, we say that Π preferentially entails α and write $\Pi \models_{\ll} \alpha$ iff

$$Min(\ll, Mod(\Pi)) \subseteq Mod(\alpha)$$

where $Mod(\phi)$ is the set of all interpretations wherein ϕ resp. all members of ϕ are true, and $Min(\ll, s)$ is the set of all $\ll$-minimal members of the set s.

This is of course similar to the principle of chronological minimization as introduced by Shoham[Sho88].

The way this definition works is that if there is some way of satisfying the axioms without discontinuity then such an interpretation is preferred. If discontinuities are necessary i.e. no interpretation without discontinuities satisfies all the axioms, then the definition prefers to have the discontinuities as late as possible, and secondarily it minimizes the set of discontinuities which do occur.

The set S of breakpoints is itself not minimized. In particular if two interpretations are identical except that one of them has some extra breakpoints in S, but all the parameters are continuous in the extra breakpoints, then those two interpretations are equally preferred. Breakpoints without any discontinuity in them are insignificant.

For example by this definition of entailment, the axioms characterizing the ball-and-shaft scenario will entail that the temperature of the ball is continuous as the ball begins going into the shaft, since there is no axiom that forces the temperature to be discontinuous there.

The reason for having continuity as a default is not only in order to deal with other, independent parameters for the same object, such as temperature, but also and maybe more importantly for dealing with multiple objects whose discontinuities occur independently. In an extended ball-and-shaft scenario there may be several shafts, and several balls falling asynchronously into those shafts. One should not need to have axioms saying that one ball moves continuously when another ball bounces against the wall in a shaft somewhere else. This is very similar to the "frame problem" but for a continuous domain.

Notice also that the preference relation only compares interpretations which are identical up to the point of comparison. One might think of having a stronger preference relation which minimizes breaksets chronologically regardless of whether parameter values are equal. However with such a stronger preference relation, the axioms for the ball and shaft scenario would logically entail that the ball starts as far to the left as possible, and moves as slowly as possible, since that postpones as much as possible the time-point for the first discontinuity. With the definition given above that problem does not arise.

6 Axioms and conclusions in the ball and shaft example

Let us illustrate the proposed method for the ball and shaft example that was introduced in section 2. The following axioms express the idealized physical laws that are being used:

1. $\Box Supp(x_b, y_b) \rightarrow \partial^2 y_b^r = 0$
2. $\Box Wall(x_b, y_b) \rightarrow \partial x_b^l = -\partial x_b^r$
3. $\Box \partial^2 x_b = 0$
4. $\Box \neg Supp(x_b, y_b) \rightarrow \partial^2 y_b = -9.81$
5. $\Box C(x_b) \wedge C(y_b)$
6. $\Box \neg Wall(x_b, y_b) \rightarrow C(\partial x_b)$

The following axioms characterize the physical configuration of the surface and the shaft:

7. $\Box a < b \wedge c < d$

8. $\Box Supp(x,y) \leftrightarrow (x < a \vee x > b) \wedge y = d \vee a \leq x \leq b \wedge y = c$

9. $\Box Wall(x,y) \leftrightarrow (x = a \vee x = b) \wedge c \leq y < d$

Finally the following axioms characterize the initial state of the system at time t_1:

10. $[t_1]\ x_b < a \wedge \partial x_b > 0 \wedge y_b = d \wedge C(\partial x_b)$

Theorems of differential calculus are not needed for determining semantic entailment, because of the chosen definition of interpretations, where parameters are assumed to be continuous with all their derivatives except in the break points.

The geometric quantities a, b, c and d are assumed to be constants with respect to time. If one does not want to have a separate syntactic type for such constants, this condition may be expressed by

11. $\Box C(a) \wedge \partial a = 0$

and similarly for b, c and d.

Section 4 above contained some of these axioms as examples of the syntax. However the common sense versions of the axioms in section 4 have been revised slightly. In axiom 2 the consequent $\partial y_b^l = \partial y_b^r$ has been dropped since the same result is obtained by default. Also axiom 1 is now expressed in terms of the second derivative in order to eliminate some non-intended model histories.

Let us now see how these axioms preferentially entail that the ball will end up on the bottom of the shaft. Consider the set of all interpretations which satisy the scenario and observation axioms (7) - (11), which also assign the same value to t_1, and which have identical histories up to then for all the parameters. This set of interpretations will then be further reduced by the requirement to satisfy the first six axioms, and to be minimal with respect to the preference relation. The remaining set is the set of (preferred) *model histories.*

In every model history the ball moves right at constant speed to the shaft, i.e. until $x_b = a$. It is clearly possible to satisfy all the axioms in such an interpretation. Therefore, although there may be other interpretations which introduce a breakpoint and a discontinuity before the ball reaches the shaft, and although such interpretations may also satisfy all the axioms, they can not be minimal with respect to the preference relation.

Also, in every model history it must be the case that y_b is constantly $= d$ as long as the ball moves along the support. For, any interpretation where the ball changes its vertical position obtains a contradiction between (1) and (4) at the timepoint where the ball leaves the support.

According to DC there exists some timepoint t_2 where $x_b = a$. Necessarily t_2 must be a breakpoint, and some parameter must be discontinuous there, since otherwise some axiom is violated. Two possibilities are of interest:

a) ∂x_b has a discontinuity and becomes negative, all other parameters are continuous.

b) $\partial^2 y_b$ has a discontinuity and changes to -9.81, all other parameters are continuous.

However the first alternative is excluded by axiom (6), so only the second possibility remains. In fact, axiom (6) only serves the purpose of ruling out the first alternative in cases like this one, and is somewhat contrary to the spirit of our general approach since we would prefer to obtain parameter continuity by default.

Anyway, in all the model histories the ball continues with constant ∂x_b. It also has continuous ∂y_b (by default) when it starts going into the shaft, but of course discontinuous $\partial^2 y_b$.

The next breakpoint will be either at the right wall or on the bottom of the shaft, whichever comes first. Either of those conditions must occur within finite time, according to DC. Also, any interpretations with breakpoints and discontinuities along the way to the right wall or bottom, will be deselected by the preference condition. Consider those interpretations where the ball hits the right wall of the shaft first.

Axiom (2) can only be satisfied if there is a breakpoint at the time-point when the ball's position satisfies the predicate for the walls of the shaft, i.e. when the ball hits the (right) wall. At that breakpoint x_b and y_b are continuous (by 5), ∂x_b reverses its sign (by 2), ∂y_b is continuous (by default), and $\partial^2 x_b$ and $\partial^2 y_b$ are constant (by 3 and 4).

The ball then bounces back and forth between the walls of the shaft, with decreasing y_b. Eventually y_b reaches the value c, at which point there must again be a breakpoint. There x_b and y_b are again continuous, ∂x_b is continuous (by default), $\partial^2 x_b$ remains 0 (by 3), and y_b becomes constantly equal to c by the same argument about second derivatives as when the ball was travelling along the upper support.

In every model history there is therefore some time after which the ball is forever at the bottom of the shaft, and in fact it bounces forever back and forth there. This was the desired "common sense" conclusion, which we however obtained using non-monotonic logic and differential equations, and while retaining full mathematical precision.

7 Discussion

The semantics and semantic entailment that have been defined here does not contain the notion of actions. In a separate paper [San88] we describe an extension of the present logic that also contains actions. The extension is non-trivial, since it requires an entailment criterium which turns out to be significantly more complex than the one used here.

The axiomatization of the ball and shaft scenario uses idealized physical laws where e.g. the ball has constant horizontal velocity and constant vertical acceleration. A more realistic axiomatization would give bounds for those parameters, instead of giving their

exact values. For example one of the "laws" could be that when the ball is unsupported, either its vertical acceleration is < -9, or its vertical velocity is negative, with a magnitude greater than some constant. Another axiom could say that when the ball bounces against the wall, its horizontal velocity has its sign reversed and its magnitude decreased. A change to such axioms makes the deductions a bit more cumbersome, but does not change the essence of what models are obtained and what is logically entailed by the axioms.

7.1 Observations and continuity

Observation axioms i.e. axioms describing the state of the system at a particular point in time, are crucial for example for stating the system's initial state, or for specifying the goal state(s) in the extension to planning. Observation axioms must always be written with a statement that the observed parameter is continuous at the time of observation, if it is, for example as in the ball-and-shaft scenario

$$[t_1]\ \partial x_b > 0 \wedge C(\partial x_b)$$

since otherwise the observation would not be enforced. Consider an interpretation where the history before t_1 arrives continously to a state such that $\partial x_b \leq 0$ at t_1. No interpretation of that form should be a model. However if it has t_1 as a breakpoint, and assigns a positive value to ∂x_b in t_1, then such an interpretation satisfies all the axioms except the one requiring ∂x_b to be continuous in t_1.

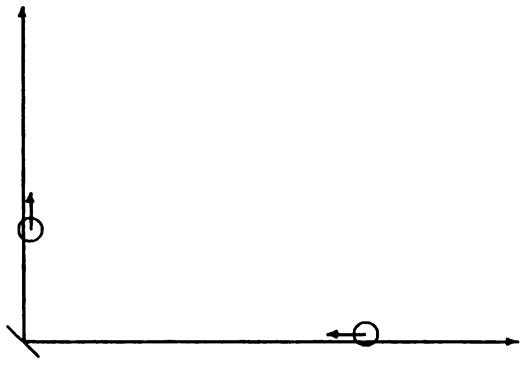
Therefore also the definition of the continuity predicate $C(\lambda)$ is that λ^l, λ, and λ^r must all be equal in the given time-point. It is not sufficient to only require that the left and right limit values be equal.

Of course if observations are recorded at a time of discontinuity for the parameter, they must specify whether they refer to the left or the right limit value.

7.2 Open and closed intervals

The main purpose of logic in the tandem of logic and DC, is to characterize the behavior of parameters around their discontinuities. To achieve that purpose we need the separate notation for referring to the left and right limit values a^l and a^r for a parameter a. This notation may seem a bit clumsy. However the problem is not a new one, and in particular one has the same problem when discrete value spaces are used. For example, if one wants to characterize the temperature of water which is brought to boil at time t_1, using a discrete value space consisting of < 100, $= 100$, and > 100 degrees Celsius, one must take care that the "value" is < 100 in the *open* interval ending in t_1, since presumably it is $= 100$ at the time t_1 itself.

Also in order to characterize with a discrete value space the x and y coordinates of a particle that travels along the x axis, and bounces in origo in order to continue along the y axis, as in the following figure:



,then it is not sufficient to only have two time intervals for before and after the bounce-time t_0. The x coordinate of the ball, x_b, is $+$ at time $< t_0$, and is $\mathbf{0}$ at time $\geq t_0$. However y_b is $\mathbf{0}$ at time $\leq t_0$ and $+$ at time $> t_0$. Therefore, in the standard approach to QR using discrete value spaces, one has to subdivide the time axis differently for characterizing x and y, or at least divide the time axis into the time-point of reflection, plus the two open intervals before and after that time-point.

Several previous authors have commented on the intricacies of dealing with parameters around the endpoints of intervals. Allen[All84] ascribed such problems to the use of time-points, and quotes it as one of the reasons why a temporal logic should be based on intervals rather than time-points.

In the perspective of earlier work, it therefore seems that the notation proposed here, i.e. using time-points but with a formula syntax that allows explicit reference to the left and right limit values of a parameter at a time-point, represents a reasonable trade-off between notational precision and notational convenience.

7.3 Naturalness of the axioms

In declarative and logic based approaches, one ideally wishes that an axiomatization should simply be a set of statements each of which comes naturally as a description of the application domain. In practice, of course one must often compromise the ideal already in order to obtain a complete (or less incomplete) axiomatization, as well as for performance reasons in an implementation when one comes to that point.

In the axioms for the ball and shaft scenario only the former consideration was involved. Most of the axioms are the "natural" ones, but the ideal has not been attained completely. Axiom (1) had to be formulated in terms of the second derivative instead of the natural formulation using the first derivative, in order to rule out certain unintended model histories as we have seen. Also the extra axiom (6) had to be introduced for that same reason.

In other words, we have approached to, but not fully arrived at the ideal situation that one could take differential equations straight from the physics textbook and use them as axioms in the logic.

7.4 Remaining mathematical groundwork

In this paper we have skipped over certain basic mathematical issues which would have to be covered in a fully rigorous account. In particular, we have assumed that minimal models always exist (no infinite sequences of more and more preferred models, except if the sequence has a limit which is even more preferred). We have also ignored whether problems could arise if there are infinitely many breakpoints.

8 Conclusion and acknowledgements

Our grand goal is to develop a coherent theory for temporal reasoning, knowledge based planning, and qualitative reasoning. The present paper has shown how the basic intentions of qualitative reasoning can be achieved within a framework of non-monotonic temporal logic. The logical next steps are to proceed to planning, if possible using the same approach or an extension of it, and also to try to represent additional physical scenarios using the same method as for the ball and shaft scenario.

The work reported here has been catalyzed by regular discussions with Lennart Ljung and his colleagues in the Division of Automatic Control of our university. The significance of our group meetings is hereby acknowledged.

References

[All84] John F. Allen. Towards a General Theory of Action and Time. *Artificial Intelligence*, Vol. 23, No. 2, 1984.

[For81] Kenneth D. Forbus. Qualitative Reasoning about Physical Processes. In *Proc. IJCAI-81*, Vancouver, B.C., 1981.

[Hay79] Patrick Hayes. The Naive Physics Manifesto. In Donald Michie, editor, *Expert Systems in the Micro Electronic Age*, Edinburgh University Press, Edinburgh, 1979.

[dKl77] Johan de Kleer. Multiple representations of knowledge in a mechanics problem-solver. In *Proc. IJCAI-77*, Cambridge, Mass, 1977.

[dKB85] Johan de Kleer and John Seeley Brown. A qualitative physics based on confluences. In Daniel G. Bobrow, editor, *Qualitative Reasoning about Physical Systems*, pages 7–83, MIT press, 1985.

[Kui82] Benjamin Kuipers. Getting the Envisionment Right. In *Proc. AAAI-82*, 1982.

[San88] Erik Sandewall. *Filter Preferential Entailment for the Logic of Action in Almost Continuous Worlds.* Technical Report, Linköping University, Dept. of Computer and Information Science, 1988.

[Sho88] Yoav Shoham. *Reasoning about Change.* MIT Press, 1988.

[Str88] Peter Struss. Mathematical aspects of qualitative reasoning. *International Journal for Artificial Intelligence in Engineering*, 1988.

Subsumption in KL-ONE is Undecidable

Manfred Schmidt-Schauß
DFKI, Universität Kaiserlautern
PO-box 3049, D-6750 Kaiserslautern
F.R. Germany

Abstract

It is shown that in the frame-based language KL-ONE it is undecidable whether one concept is subsumed by another concept. In fact a rather small sublanguage of KL-ONE called $\mathcal{ALR}$ is used which has only the concept forming operators conjunction, value restriction, and role value maps using only '='. In particular, number restrictions are not used. The language $\mathcal{ALR}$ can be viewed as an extension of feature terms without complements and unions, where features have sets as values instead of elements. Our result shows that there is a basic difference between feature terms and KL-ONE, since the complexity of subsumption switches from quasi-linear to undecidable if the restriction is dropped that roles are functional.

1. Introduction

The knowledge representation language KL-ONE [Brachman and Schmolze, 1985, Brachman et. al., 1985, Schmolze and Israel, 1983] permits describing concepts on the basis of unary predicates (concepts, frames) and binary predicates (roles, slots). It is a language supporting the definition and exploitation of taxonomical knowledge including multiple inheritance, and has as advantage a declarative semantics that permits to compare implementations. Several KL-ONE-related languages are currently being used as the basis of knowledge representation systems [Brachman and Schmolze, 1985, von Luck et. al. 1987, Kaczmarek et. al., 1986, Mac Gregor, 1988], in particular for natural language processing.

The terminological component (T-Box) of KL-ONE gives a potential user the possibility to structure a domain of interest by using concepts and roles (frames and slots). Usually, such a description starts by postulating certain primitive concepts and roles, and afterwards defining concepts using the operators available. An example for such concepts may be person, animal, female, male, thing, and examples for roles are child (has-child) and father (father-of). New concepts can be defined via conjunction ("a person that is female"), via value restriction with respect to some role ("a person with every male friend is a doctor"), via number restriction ("a person with more than three children"), via role value maps ("a man with every child of a child of his father is also a child"), and some other formation possibilities. It is also possible to restrict models by some axioms, for example by requiring that certain concepts are disjoint or that some concepts form a covering of another concept.

Research in computational linguistics has lead to a related knowledge representation method as the basis of unification grammars [Kay, 1985, Shieber, 1986], also called feature structures. H. Aït-Kaci has independently developed a very similar mechanism [Aït-Kaci, 1984, Aït-Kaci and Nasr, 1986] with the intention to apply it in knowledge representation systems. The so-called Ψ-terms are defined and implemented in a Logic Programming language. A similar device are the feature terms, described in [Smolka and Aït-Kaci, 1987, Smolka, 1988], where concepts are called sorts and roles are called features. We give an example for feature terms: Let **car** be a sort symbol (concept) and let *speed* and *age* be features applicable to cars. Then (**car** $\sqcap$ (*age* $\downarrow$ *speed*)) is a feature term denoting the set of all cars that have equal age and

speed. A feature term used in linguistic application may be (**sentence** ⊓ ((*subj number*) ↓ (*pred number*))), which denotes sentences with agreeing number of subject and number of predicate. The difference to knowledge representation á la KL-ONE is that features denote partial functions. The term forming rules for feature terms directly correspond to the concept forming possibilites conjunction, value restriction and role-value maps available in KL-ONE. In addition, feature terms may be defined using a predefined lattice structure on sorts (the primitive concepts). Indeed, the expressiveness of feature terms can be viewed as a subset of KL-ONE where all roles are functional.

The basic service provided by all these languages is a reasoning facility called 'classification' that informs the user whether one of his already defined concepts is subsumed by some other already defined concept. H. Levesque and R. Brachman strongly argue in several papers [Brachman and Levesque, 1984, Levesque and Brachman, 1985, Levesque and Brachman, 1987] that classification should be executable in polynomial time. They showed that there is a tradeoff between expressiveness of the concept description language and the complexity of the subsumption test. In particular they proved that reasonably expressive concept description languages have a co-NP-hard subsumption problem.

In the feature term language the main inference is called 'unification'. Basically, unification of two feature terms amounts to computing a simplified representation of their conjunction and a test, whether this conjunction denotes a nonempty set. The test for consistency is in fact a classification problem. It was shown by H. Aït-Kaci [1984], that classification can be performed in quasi-linear time. Recently it was shown that the classification problem for feature terms including negation is co-NP-complete [Smolka, 1988].

In my opinion, knowledge representation formalisms that don't admit a polynomial classification algorithm are nevertheless useful, since for example most NP-complete problems permit algorithms that require polynomial time in average. However, a formalism with an undecidable subsumption is unsatisfactory, since in this case it may even be the case that consistency of concepts is not recursively enumerable. Using a formalism with undecidable subsumption in practice means that the corresponding classification is either incomplete or that further constraints to restrict concept formation are to be employed. If the latter is the case, then these restriction should show up in the theory, possibly yielding a decidable subsumption for the restricted language. A further argument for decidability of subsumption is that only in this case it is sensible to combine such a formalism with a first-order language (A-Box) with the intention to exploit the mechanisms and algorithms of the knowledge representation formalism, since otherwise the T-Box is already as powerful as any programming language. However, there are some recent approaches using feature terms as constraints [Höhfeld and Smolka, 1988, Bläsius and Hedtstück, 1988]. These methods use a lazy classification and can thus tolerate undecidable subproblems by postponing the decision until further information is available. Both approaches are restricted to feature terms, however, an extension to KL-ONE-like concept terms appears possible.

Although there is agreement that role value maps provide complications, there are systems allowing them as descriptive possibility and use them also in the (incomplete) classification algorithm. For example NIKL [Kaczmarek et. al., 1986] admits the full expressive power of role value maps, whereas for example in BACK [von Luck et. al., 1987] role value maps are strongly restricted. There are several knowledge representation systems using feature terms and role value maps in the restricted form of agreements. Since for feature terms it is known that the consistency check is quasilinear, and co-NP-complete if complements are permitted, the demand which suggests itself is to extend features to arbitrary binary predicates (or roles).

In this paper it is shown that subsumption in $\mathcal{ALR}$, a rather small sublanguage of KL-ONE is undecidable. This result was a surprise, since the very similar language of feature terms has a tractable classification problem, and my and other peoples expectation was that extending feature terms by set-valued features would only moderately increase the complexity of classification. This shows that role value maps in the restricted form are acceptable, but that their general form as used in KL-ONE should be restricted.

For this result, conjunction, value restriction, and role value maps using only '=' are needed, but not more. The proof uses the undecidability of the word problem in groups. This is more convenient than the word problem in semigroups, since using the latter one can only show undecidability of subsumption if '$\subseteq$' is permitted as comparison operator in role value maps.

Recently, P. F. Patel-Schneider [1989] has shown that classification in NIKL is undecidable. His sublanguage of KL-ONE requires more of the expressiveness of KL-ONE than our sublanguage, such as role value maps with '$\subseteq$' as comparison operator, inverse roles and number restrictions. In the paper it is mentioned that inverse roles can be omitted, but role value maps with '$\subseteq$' seem to be indispensible. K. Schild [Schild, 1988] presents a proof showing that in a in a KL-ONE-related language that supports only role-definition subsumption of roles is undecidable. The language permits the definition of complements of roles, composition of roles and conjunction of roles.

The paper is structured as follows: In section 2 the syntax and semantics of $\mathcal{ALR}$ and of $\mathcal{ALRC}$ are given. In section 3 we give the proof of undecidablity of subsumption by reducing the word problem in groups to it and furthermore discuss some consequences.

2. The Language $\mathcal{ALR}$

In the following we describe the syntax and semantics of the sublanguage $\mathcal{ALR}$ of the terminological language KL-ONE [Brachman and Schmolze, 1985, Schmolze and Israel, 1983] as well as of an extension $\mathcal{ALRC}$ of $\mathcal{ALR}$ in a slightly modified linear syntax, but the same semantics. Our language $\mathcal{ALR}$ allows to construct concept expressions by the constructors given below, but it does not support the definition of roles. Consequently we are mainly interested in concepts and their relations.

There are disjoint sets of role symbols and concept symbols. Concept expressions are:

i) concept symbols
ii) $C \sqcap D$, if C and D are concept expressions.
iii) $\forall R{:}C$, if R is a role symbol and C is a concept expression.
iv) $P = Q$ if P and Q are lists of roles.

Concept expressions in $\mathcal{ALR}$ are composed from symbols and the three methods of construction, possibly by inserting some brackets when necessary. The concept expression for the concept "a man with every child of a child of his father is also a child" would be something like (man $\sqcap$ ((father, child, child) = (child))). In the NIKL-syntax this would be (and man (all (compose father child child) child) (all child (compose father child child) child)).

In the proofs we will use the more expressive language $\mathcal{ALRC}$ that extends the language $\mathcal{ALR}$ by the following three formation rules for concepts:

v) $C \sqcup D$, if C and D are concept expressions.
vi) $\exists R{:}C$, if R is a role symbol and C is a concept expression.
vii) $\neg C$ if C is a concept expression.

The language $\mathcal{ALRC}$ is an extension of the attributive concept description language $\mathcal{ALC}$ [Schmidt-Schauß and Smolka, 1988] by role value maps.

We need some facts about relations, compositions of relations and applications of relations to sets and elements. Let R,S be relations over a set M, i.e., $R,S \subseteq M \times M$.

The composition of R and S is defined as:

$R \circ S := \{(x,y) \mid \exists z \in M\ (x,z) \in R \wedge (z,y) \in S\}$.

The application of a relation R to a set $s \subseteq M$ is defined as follows:

$sR := \{y \mid \exists x{:}\ x \in s \wedge (x,y) \in R\}$.

The application of R to an element x is defined analogously:

$xR := \{y \mid (x,y) \in R\}$.

Obviously, we have

$s(R \circ S) = (sR)S$ and $x(R \circ S) = (xR)S$

for a set s and an element x.

The semantics of the roles, concept symbols and concept expressions is as usual [Levesque and Brachman, 1985, Levesque and Brachman, 1987, von Luck et. al., 1987]. We give the semantics of the more general language $\mathcal{ALRC}$, which includes the semantics of $\mathcal{ALR}$.

An **interpretation** I is a pair (M, I), where M is a set and I an interpretation function, such that

i) for every concept symbol C: $I(C) \subseteq M$
ii) for every role symbol R: $I(R) \subseteq M \times M$.

Lists of roles are interpreted as the composition of relations:

$I((R_1,\ldots,R_n)) = I(R_1)\circ\ldots\circ I(R_n)$.

We intepret defined concepts as subsets of M as follows:

$I(C \sqcap D) = I(C) \cap I(D)$
$I(C \sqcup D) = I(C) \cup I(D)$
$I(\forall R{:}C) = \{x \in M \mid \forall y{:}\ (x,y) \in I(R) \Rightarrow y \in I(C)\}$
$I(\exists R{:}C) = \{x \in M \mid \exists y{:}\ (x,y) \in I(R) \wedge y \in I(C)\}$.
$I(P = Q) = \{x \in M \mid x(I(P)) = x(I(Q))\}$
$I(\neg C) = M - I(C)$,

where C and D are concepts, R is a role symbol, and P and Q are lists of roles.

Subsumption and Consistency are defined with respect to this semantics:

- A concept expression C **subsumes** a concept expression D, iff for all interpretations I, we have $I(C) \supseteq I(D)$.
- A concept C is **consistent**, iff there exists an interpretation I, such that $I(C) \neq \emptyset$, otherwise C is called **inconsistent**.
- Two concepts C and D are **equivalent**, iff C subsumes D and D subsumes C.

Since we have negation in $\mathcal{ALRC}$, and since the empty concept can be defined, subsumption problems in $\mathcal{ALRC}$ are equivalent to inconsistency problems.

2.1 Lemma. Let C, D be concept expressions with respect to $\mathcal{ALRC}$. Then the following three statements are equivalent:

i) D subsumes C,
ii) $\neg D \sqcap C$ is an inconsistent concept.
iii) $\neg D \sqcap C$ is subsumed by $\neg D \sqcap D$ (the empty concept).

Proof. The concept D subsumes C, iff $I(D) \supseteq I(C)$. This in turn is equivalent to $(M - I(D)) \cap I(C) = \emptyset$. Thus i) is equivalent to ii). Obviously, $I(\neg D \sqcap D) = \emptyset$ for all interpretations I, hence ii) is equivalent to iii). ■

Subsumption is not equivalent to consistency of concepts in the language $\mathcal{ALR}$, since in $\mathcal{ALR}$ all concepts are consistent. Even if $\exists R{:}C$ and the role-defining operator (`restrict` R C) from NIKL are permitted, then all concepts remain consistent, which follows by considering a one-element interpretation $I = (M,I)$ with $M = \{m\}$, where all concept symbols are interpreted as M and all roles as $\{(m, m)\}$. In this interpretation, the restriction of a role has no effect, and all defined concepts have M as extension. Thus all concepts remain consistent.

Let us compare the expressiveness of $\mathcal{ALR}$ and $\mathcal{ALRC}$ with that other knowledge representation languages:

Obviously, $\mathcal{ALR}$ is a sublanguage of KL-ONE. $\mathcal{ALR}$ has less expressiveness than the sublanguage of NIKL used by P.F.Patel-Schneider [1989] to show undecidability of subsumption in NIKL. The $\mathcal{ALR}$-concept $\forall R{:}C$ is equivalent to (`all` R (`restrict` R C)) in NIKL and to (`all` R C) in $\mathcal{FL}$ [Levesque and Brachman, 1985]. A concept description using role value maps $((R_1, \ldots, R_n) = (S_1, \ldots, S_m))$ is equivalent to the NIKL-expression (`and` (`all` (`compose` $R_1, \ldots, R_n$) (`compose` $S_1, \ldots, S_m$) (`all` (`compose` $S_1, \ldots, S_m$) (`compose` $R_1, \ldots, R_n$))). I conjecture that the expressiveness of $\mathcal{ALR}$ is strictly smaller, since it is not possible to define functional roles in $\mathcal{ALR}$.

There is a close relationship between $\mathcal{ALR}$ and feature terms. $\mathcal{ALR}$ can be seen as feature terms (without negation and union) with the semantics of features changed from (partial) functional to arbitrary binary relations. The same relationship holds between $\mathcal{ALRC}$ and feature terms with complements and union. Of course, the descriptive power is not really comparable, since for example the rules for computing with feature terms and concept expressions in $\mathcal{ALRC}$ are different [Smolka, 1988, Schmidt-Schauß and Smolka, 1988]. For example the rule $\forall R{:}(C \sqcup D) \rightarrow (\forall R{:}C) \sqcup (\forall R{:}D)$ is valid for feature terms, whereas this is false in $\mathcal{ALRC}$.

3. Subsumption in $\mathcal{ALR}$ is Undecidable.

In this section we will show that subsumption in $\mathcal{ALR}$ is undecidable by reducing the word-problem in groups to it. Since the semantics of $\mathcal{ALRC}$ and $\mathcal{ALR}$ are compatible, it is sufficient to show that subsumption of $\mathcal{ALR}$-expressible concept expressions is undecidable in $\mathcal{ALRC}$. Hence we will use the language $\mathcal{ALRC}$ in the following, but we will encode the subsumption problem using operators from $\mathcal{ALR}$.

Similar as in [Schmidt-Schauß and Smolka, 1988, Smolka, 1988], we transform subsumption problems of $\mathcal{ALR}$-concepts into a system C of constraints, where every

single constraint is of one of the forms $s \subseteq C$, $x \in C$, $x \in s$, $s = t$. We write x, y, z for element variables, s,t for expressions of the form $xR_1 \ldots R_n$, and C, D for concept expressions. The reason for using constraints is that the proofs are far more readable than in linear syntax manipulating concept expressions and that subsumption algorithms can be described in an elegant way (cf. [Schmidt-Schauß and Smolka, 1988]).

Let $I = (M, I)$ be an interpretation. Let α be an assignment of elements in M to element variables of $\mathcal{C}$. We assume that α extends the interpretation function I, i.e., $\alpha C = I(C)$, $\alpha(xP) := (\alpha x)I(P)$ for a concept C and a list of roles P.

Then we say α **satisfies** $\mathcal{C}$, if the following holds:

for $(x \in A) \in \mathcal{C}$, we have $\alpha x \in \alpha A$

for $(A \subseteq B) \in \mathcal{C}$, we have $\alpha(A) \subseteq \alpha B$

for $(A = B) \in \mathcal{C}$, we have $\alpha A = \alpha B$.

A constraint system $\mathcal{C}$ is **consistent**, iff there exists an interpretation I and an assignment α with respect to this interpretation such that α satisfies $\mathcal{C}$. Otherwise $\mathcal{C}$ is called **inconsistent**.

3.1 Lemma. Let C be a concept expression.

Then C is consistent, iff the constraint system $\{x \in C\}$ is consistent.■

There are several rules for replacing concepts and constraints which make life easier, and which preserve consistency and inconsistency of constraint systems (cf. [Schmidt-Schauß and Smolka, 1988, Smolka, 1988]:

$\neg(\forall R{:}C)$	$\leftrightarrow$	$\exists R{:}\neg C$
$\neg(\exists R{:}C)$	$\leftrightarrow$	$\forall R{:}\neg C$
$x \in A \sqcap B$	$\leftrightarrow$	$x \in A, x \in B$
$x \in \forall R{:}C$	$\leftrightarrow$	$xR \subseteq C$
$x \in \exists R{:}C$	$\leftrightarrow$	$y \in xR, y \in C$ where y is a new variable
$x \in (P = Q)$	$\leftrightarrow$	$xP = xQ$

In order to show undecidability of subsumption, we use the undecidability of the word problem in groups [Boone, 1959, Novikov, 1955, Stillwell, 1982]. Such a problem looks as follows: Let $R_1,\ldots,R_{n'}$ be the symbols of a group and let $P_1=Q_1,\ldots,P_{m'}=Q_{m'}$ be the generating relations of some group. Then the word problem is to test given strings P and Q, whether $P = Q$ is derivable from these relations and the axioms for a group. Our aim is to show that there is a subsumption problem that is equivalent to this word problem.

In order to avoid clumsy notation for the used groups, we assume that only associativity is built-in and that the semi-group-defining relations imply that the semigroup defined by the relations is a group. If we have given generating relations under the assumption that all axioms for a group are built-in, the following procedure gives relations for semigroups ensuring that the generated semigroup is an isomorphic group: Add a new symbol R_e standing for the unit, and add for every symbol R_i a new symbol R_i^- for the inverse. Then add the relations: $R_e \cdot R = R$ and $R \cdot R_e = R$ for every symbol $R \in \{R_e, R_i, R_i^-, i = 1,\ldots,n'\}$ and add the relations $R_i^- \cdot R_i = R_e$ and $R_i \cdot R_i^- = R_e$ for all symbols R_i. The defining relations for the group are translated as follows: We assume that the words occurring in the relations are composed of symbols or inverses of symbols. The unit is translated into R_e, symbols are translated identically and inverses of symbols are translated by $(R_i)^{-1} \rightarrow R_i^-$. Now the new relations together with the translated generated relations defining the group provide a semigroup, which is isomorphic to the original group and has an equivalent word problem.

These considerations permit us to assume in the following that the symbols are $R_1,\ldots,R_n$ and that the relations are $P_1=Q_1,\ldots,P_m=Q_m$, where the group defining relations are among these relations. By $\mathcal{G}$ we denote the free semigroup (which is in fact a group) generated by the symbols $R_1,\ldots,R_n$ and the relations $P_1=Q_1,\ldots,P_m=Q_m$. This is the semigroup consisting of all congruence-classes of words from $\{R_1,\ldots,R_n\}^*$, where two words are congruent, if the least congruence on $\{R_1,\ldots,R_n\}^*$ that contains the relations makes them congruent. For convenience we denote elements of $\mathcal{G}$ by [P], where P is a string of symbols and [P] denotes the congruence class with respect to the defining relations.

We say an equation $P = Q$ is derivable from the relations, denoted as $\vdash P = Q$, if it can be generated from the relations using the following rules:

1) $\vdash P_i = Q_i$ for all generating relations.
2) $\vdash P = P$ for all words P from $\{R_1,\ldots,R_n\}^*$.
3) $\vdash P = Q$ if $\vdash Q = P$
4) $\vdash P = Q$ if $\vdash P = P'$ and $\vdash P' = Q$
5) $\vdash PP' = QQ'$ if $\vdash P = Q$ and $\vdash P' = Q'$.

We have [P] = [Q] iff the equation P = Q can be derived from the relations using rules 1) – 5) [Burris and Sankappanavar, 1981, Grätzer, 1979]. Note that this means that for two given words P and Q it is semidecidable, whether they are congruent by using the calculus above for enumeration of all derivable equations.

As an example consider the integers, which form a group with respect to addition. We use the usual notation of 0,1,-1 and + instead of R_e,R_1, R_1^- and "•". If all axioms of a group are built-in, then "1" is sufficient as symbol and there are no relations. Considered as semigroup, the group of integers is generated by the three symbols 0,1,-1 and the relations 0+1 = 1, 1+0 = 1, 1+1 = 1+1, 0+(-1) = -1, (-1)+0 = -1, (-1)+(-1) = (-1)+(-1), 1+(-1) = 0, (-1)+1 = 0. A derivable equation is for example 1+(-1) = (-1)+1.

In the following we sometimes use the symbols R_i, i = 1,...,n from above also as role symbols. We need an additional role symbol R that is not among these role symbols. Furthermore we use P_i, Q_i, P, and Q also for the lists of roles. Thus it depends on the context whether P or Q is meant as a word in the group or as a list of role symbols.

Now we define several concepts that are needed for the subsumption problem encoding the word problem. For convenience we use ⊓ as associative operator and write lists of roles as composition.

$C_1 := (R \bullet R_1 = R) \sqcap \ldots \sqcap (R \bullet R_n = R)$

$C_2 := \forall R:(P_1 = Q_1) \sqcap \ldots \sqcap \forall R:(P_m = Q_m)$

Now let $C := C_1 \sqcap C_2$ and let $D_{P,Q} := \forall R: (P=Q)$.

The subsumption problem, which we are interested in is whether $D_{P,Q}$ subsumes C.

The idea of the construction is to view the relations in the role value maps as relations that define a semigroup and then to make deductions using the deduction rules 1) – 5) above. The concept C_2 encodes the relations that are used to define a particular group, whereas C_1 is only technical; it encodes some fixed point properties that will guarantee the correctness of deducing new role value maps from the given ones similar to deducing new equations form given equations.

Let $C_{\mathcal{G}}$ be the following constraint (coming from C):

$$C_{\mathcal{G}} := \{ xR \bullet R_1 = xR, \ldots, xR \bullet R_n = xR, \; xR \subseteq (P_1 = Q_1), \ldots, xR \subseteq (P_m = Q_m)\},$$

and let $C_{\mathcal{G}}(P,Q)$ be the following constraint system (coming from $\neg D_{P,Q} \sqcap C$):

$C_{\mathcal{G}}(P,Q) := \{y \in xR, y \in \neg(P = Q)\} \cup C_{\mathcal{G}}$.

Now we can prove the main result as a sequence of lemmas.

3.2 Lemma. $D_{P,Q}$ subsumes C, iff the constraint system $C_{\mathcal{G}}(P,Q)$ is inconsistent:

Proof. Lemma 2.1 yields that $D_{P,Q}$ subsumes C iff $(\neg D_{P,Q}) \sqcap C$ is an inconsistent concept, and Lemma 3.1 yields that this is equivalent to the inconsistency of the constraint system $\{x \in (\neg D_{P,Q}) \sqcap C\}$. Due to the rules above, we can transform this constraint system in several steps as follows:

$\{x \in \neg(\forall R: (P=Q)), x \in C_1 \sqcap C_2\}$

$\Leftrightarrow \quad \{x \in \exists R:\neg(P=Q), x \in C_1, x \in C_2\}$

$\Leftrightarrow \quad \{y \in xR, y \in \neg(P = Q), x \in C_1, x \in C_2\}$.

If we develop the concepts C_1 and C_2, then we obtain the constraint system:

$\{ y \in xR, y \in \neg(P = Q), \; xR \bullet R_1 = xR, \ldots, xR \bullet R_n = xR, \; xR \subseteq (P_1 = Q_1), \ldots, xR \subseteq (P_m = Q_m)\}$,

which is exactly the above defined system $C_{\mathcal{G}}(P,Q)$. Since all used transformations preserve consistency and inconsistency, the lemma holds. ■

Now we can prove the crucial fact that the deduction rules above can be simulated in the constraint system $C_{\mathcal{G}}(P,Q)$. The basic idea is the use of the additional constraints $xR \bullet R_i = xR$, which permit to view lists of roles in the constraints as words in groups. Without this "technical" addition, this is impossible.

3.3 Lemma. If P' = Q' is derivable from the relations defining $\mathcal{G}$, then for every constraint system C_0 with $C_{\mathcal{G}} \subseteq C_0$, there exists a constraint system $C_{\mathcal{G}}'$ with the following properties:

i) $C_0 \subseteq C_{\mathcal{G}}'$

ii) $\{xR \subseteq (P' = Q')\} \subseteq C_{\mathcal{G}}'$

iii) C_0 is consistent iff $C_{\mathcal{G}}'$ is consistent.

Proof. For a constraint system C let EQ(C) denote the set $\{P = Q \mid (xR \subseteq (P = Q)) \in C\}$.

We prove by induction on the length of derivations using the rules 1) – 5) above that given a constraint

system $\mathcal{C}$ with $\mathcal{C}_0 \subseteq \mathcal{C}$ and an equation P' = Q' derivable from EQ($\mathcal{C}$), the constraint system $\{xR \subseteq (P' = Q')\} \cup \mathcal{C}$ is consistent, iff $\mathcal{C}$ is consistent. That the consistency of $\{xR \subseteq (P' = Q')\} \cup \mathcal{C}$ implies the consistency of $\mathcal{C}$, is obvious and hence not mentioned in the following.

Let $\mathcal{C}$ be a consistent constraint system containing $\mathcal{C}_0$.

1) If P' = Q' is derivable from EQ($\mathcal{C}$) with rule 1), then $xR \subseteq (P' = Q')$ is a constraint in $\mathcal{C}_0$, since $\mathcal{C}_0$ contains $\mathcal{C}_\mathcal{G}$.
2) For every word P from $\{R_1,\ldots,R_n\}^*$, the constraint system $\mathcal{C} \cup \{xR \subseteq (P = P)\}$ is consistent, since every assignment α satisfies $xR \subseteq (P = P)$.
3) For an equation P = Q in EQ($\mathcal{C}$), it is obvious that $\mathcal{C} \cup \{xR \subseteq (Q = P)\}$ is consistent, since for every assignment α, we have $\alpha(P = Q) = \alpha(Q = P)$.
4) Let P = P' and P' = Q be in EQ($\mathcal{C}$) and let α be an assignment that satisfies $\mathcal{C}$. This means $\alpha(xR) \subseteq \alpha(P = P')$ and $\alpha(xR) \subseteq \alpha(P' = Q)$. By the semantics we have that for every element $a \in \alpha(xR)$: $a(\alpha P) = a(\alpha P')$ and $a(\alpha P') = a(\alpha Q)$. Hence we have also $a(\alpha P) = a(\alpha Q)$, hence $\alpha(xR) \subseteq \alpha(P = Q)$ holds. Thus α satisfies the constraint system $\mathcal{C} \cup \{xR \subseteq (Q = P)\}$, hence it is consistent.
5) Let P = Q and P' = Q' be in EQ($\mathcal{C}$) and let α be an assignment that satisfies $\mathcal{C}$. This means $\alpha(xR) \subseteq \alpha(P = Q)$ and $\alpha(xR) \subseteq \alpha(P' = Q')$. Now the constraints from C_1 have to be used! Since $\alpha(xR \circ R_i) = \alpha(xR)$ for all i=1,...,n, we have also $\alpha(xR \circ P) = \alpha(xR) = \alpha(xR \circ Q)$ by repeated application. For every $a \in \alpha(xR)$ the equation $a(\alpha P) = a(\alpha Q)$ holds. Furthermore $a(\alpha P) \subseteq \alpha(xR)$. Hence for every element $b \in a(\alpha P)$, we have $b(\alpha P') = b(\alpha Q')$. If we take the union of all sets $b(\alpha P')$ and $b(\alpha Q')$ where b ranges over the whole set $a(\alpha P)$, we obtain that $a(\alpha(P \circ P') = a(\alpha Q \circ Q')$ for every $a \in \alpha(xR)$. Hence $\alpha(xR) \subseteq \alpha(P \circ P' = Q \circ Q')$ holds. Thus α is an assignment that satisfies the constraint system $\mathcal{C} \cup \{xR \subseteq (P \circ P' = Q \circ Q')\}$, hence this constraint system is consistent. ■

3.4 Lemma. Let P and Q be words over $\{R_1,\ldots,R_n\}^*$. Then [P] = [Q] iff $\mathcal{C}_\mathcal{G}(P,Q)$ is inconsistent.

Proof.

"⇒": Assume by contradiction that $\mathcal{C}_\mathcal{G}(P,Q)$ is consistent. If [P] = [Q] holds in $\mathcal{G}$, then the rules 1) - 5) are sufficient to derive P = Q from the relations. Lemma 3.3 shows that there exists a consistent constraint system $\mathcal{C}_\mathcal{G}'$ containing $\mathcal{C}_\mathcal{G}(P,Q) \cup \{xR \subseteq (P = Q)\}$. The system $\mathcal{C}_\mathcal{G}(P,Q)$ contains the constraints $y \in xR$, $y \in \neg(P = Q)$, which contradict the constraint $\{xR \subseteq (P = Q)$. Hence $\mathcal{C}_\mathcal{G}(P,Q)$ is inconsistent.

"⇐": We show that if [P] ≠ [Q], then $\mathcal{C}_\mathcal{G}(P,Q)$ is consistent. Therefore we assume that [P] ≠ [Q]. An interpretation $I = (I, M)$ that satisfies $\mathcal{C}_\mathcal{G}(P,Q)$ can be constructed as follows: Let the domain M be $M := \{a\} \cup \mathcal{G}$, where $a \notin \mathcal{G}$, let $I(R) := \{(a,g) \mid g \in \mathcal{G}\}$, and let $I(R_i) := \{(g, g \circ [R_i]) \mid g \in \mathcal{G}\}$ for i = 1,...,n. The assignment α is defined such that $\alpha x := a$. This means that $\alpha(xR) = \mathcal{G}$. Since multiplication from right in a group is a bijection, the constraints $xR \circ R_1 = xR$, ... , $xR \circ R_n = xR$ are satisfied. The constraints $xR \subseteq (P_1 = Q_1)$, ..., $xR \subseteq (P_m = Q_m)$ are also satisfied, since the equations $[P_i] = [Q_i]$, i = 1,...,m hold in $\mathcal{G}$. It remains to be shown that $y \in \neg(P = Q)$ and $y \in xR$ can be satisfied. The assigment of the unit $1_\mathcal{G}$ in $\mathcal{G}$ to y, i.e., $\alpha y := 1_\mathcal{G}$ gives an element that is contained in the sets $\alpha(xR)$ and $\alpha(\neg(P = Q))$, since [P] ≠ [Q] in $\mathcal{G}$. Hence the thus defined assignment α satisfies the constraint $\mathcal{C}_\mathcal{G}(P,Q)$. ■

3.5 Theorem. Subsumption in $\mathcal{ALR}$ and hence in KL-ONE, is undecidable.

Proof. Obviously the subsumption problem in Lemma 3.2 can be formulated in $\mathcal{ALR}$, Lemmas 3.2, 3.3, 3.4, show that the concepts $D_{P,Q}$ subsumes C, iff [P] = [Q] with respect to the group defined by the relations. Now the well-known result that the word problem in groups is undecidable [Boone, 1959, Novikov, 1955, Stillwell, 1982] implies that subsumption is undecidable. ■

Note that the reason for using the undecidability of the word problem in groups rather than in semigroups or monoids is that in the proof of Lemma 3.4, the constraints $xR \circ R_i = xR$ have to be satisfied, which requires that multipliciation from right must be surjective.

The result in [Boone, 1959, Novikov, 1955, Stillwell, 1982] shows that there exists a group, such that the word

problem in this group is undecidable. In our context this means, that there exists a fixed concept C such that it is undecidable whether a given concept $D_{P,Q}$ subsumes C.

3.6 Corollary. In $\mathcal{ALR}$ there exists a fixed concept C, such that it is undecidable, whether a given concept D subsumes C.

As a further corollary of Theorem 3.5 we obtain also that some problems cannot be recursively enumerable, such as non-subsumption in $\mathcal{ALR}$ and consistency of concepts in the language $\mathcal{ALRC}$.

3.7 Corollary. Consistency of concepts in $\mathcal{ALRC}$ is not recursively enumerable.

Proof. Assume for contradiction that the consistent concepts of $\mathcal{ALRC}$ can be recursively enumerated. Since the calculus consisting of the 5 rules for the equations in groups is complete, it follows from Lemma 3.4 that the inconsistent constraints $C_G(P,Q)$ can be recursively enumerated. Using Lemma 3.1 the assumption that consistent concepts of $\mathcal{ALRC}$ are recursively enumerable implies that the consistent constraint systems $C_G(P,Q)$ can be recursively enumerated, since $C_G(P,Q)$ is equivalent to $\{x \in \neg D_{P,Q} \sqcap C\}$. This means inconsistency of $C_G(P,Q)$ is decidable, which contradicts Lemma 3.4 and Theorem 3.5. ■

Of course, this does not hold for $\mathcal{ALR}$ alone, since all $\mathcal{ALR}$-concepts are consistent.

Corollary 3.7 has as a curious consequence that we can give a nonconstructive proof that there must be a consistent concept that denotes the empty set in all finite models:

3.8 Corollary. There exists a consistent $\mathcal{ALRC}$-concept C, such that C denotes a nonempty set only in infinite interpretations.

Proof. Assume, that the corollary is false. Then for every consistent $\mathcal{ALRC}$-concept C, there exists a finite interpretation, such that C is interpreted as a nonempty set. This implies that consistency of concepts would be recursively enumarable, which contradicts Corollary 3.7. ■

In the following we describe some consequences for languages that extend the feature term languages in somme way. Note that our proof of undecidability of subsumption in $\mathcal{ALR}$ does not work for the simple feature term language, since R must be interpreted as a proper role, which is not possible in the feature term language.

Let us define the language $\mathcal{ALR}^*$ as an extension of the feature term language. The language $\mathcal{ALR}^*$ has (disjoint) sets of role and feature symbols. The interpretation of roles is as usual, whereas the interpretation of a features F should be partial functions, i.e.,

$$(a,b) \in I(F) \wedge (a,c) \in I(F) \Rightarrow b = c.$$

Concept expressions in $\mathcal{ALR}^*$ are:

i)	concept symbols	
ii)	$C \sqcap D$,	if C and D are concept expressions.
iii)	$\forall R:C$,	if R is a role symbol and C is a concept expression.
iv)	$F:C$,	if F is a feature symbol and C is a concept expression.
v)	$P = Q$	if P and Q are lists of roles or features, where the first element may be a role, whereas all other elements in the list are features.

The semantics of concept expressions is slightly changed, the conjunction is (as usual) interpreted as intersection, but the constructs F:C and P=Q include definedness of roles and features. Let $I = (M, I)$ be an interpretation, then

$$I(F:C) := \{a \in M \mid \exists b \in I(C): (a, b) \in I(F)\}$$

$$I(P=Q) := \{a \in M \mid (\exists b: b \in I(C): (a, b) \in I(P)) \wedge al(P) = al(Q)\}$$

The language $\mathcal{ALR}^*$ is a slight extension of the (simple) feature term language. Nevertheless, subsumption is undecidable:

3.9 Theorem. Subsumption in $\mathcal{ALR}^*$ is undecidable.

Proof. The proofs of the lemmas 3.1 and 3.2 remain valid. In the proof of Lemma 3.3 one has to take into account that the semantics has slightly changed. For 1) - 4) there are no problems. In the proof of 5) there are some additons: On has to prove that the assignment α has the additional property that $\alpha(P \circ P' = Q \circ Q') \neq \emptyset$, which holds, since the $\alpha(xR)$ is not empty. The proof of Lemma 3.4 holds also for the modified semantics without changes, since the interpretations for the roles R_i are partial functions in the model construction part.

Finally Theorem 3.5 with the modified Lemmas 3.1 – 3.4) can be applied and yields that subsumption in $\mathcal{ALR}^*$ is undecidable. ■

An extension of the feature term language which should be investigated is the simple feature term language, where roles are added, but not permitted , i.e., i) - iv) are as above, but v) is slightly changed:

v') P = Q if P and Q are lists of features.

I concjecture that subsumption in this language remains decidable.

4. Conclusion

We have shown that subsumption of concepts in $\mathcal{ALR}$, a considerable small sublanguage of KL-ONE, is undecidable, if the usual standard semantics is used. The reason for this undecidability result seems to be the expressive power of role value maps. They are rather intuitive at first glance, for example they allow the definition of *grand-father* in terms of the roles *father* and *mother*, but provide the full power of a programming language if used excessively.

As mentioned above, subsumption of feature terms is quasi-linear or co-NP-complete, depending on the expressiveness [Aït-Kaci, 1984, Aït-Kaci and Nasr, 1986, Smolka, 1988, Smolka and Aït-Kaci, 1987]. It is remarkable that role value maps in the feature term language do not have such a dramatic effect as in $\mathcal{ALR}$. The main difference between $\mathcal{ALR}$ and the feature term language is that roles in the latter always are functional.

There are several methods to either overcome the problem of undecidability or deal with undecidability.

A first ad-hoc possibility to re-establish decidability is to restrict the expressive power of $\mathcal{ALR}$ or $\mathcal{ALRC}$ by discarding role value maps. A language called $\mathcal{ALC}$, that allows complements in addition but no role value maps has been investigated in [Schmidt-Schauß and Smolka, 1988], where it is shown that subsumption becomes PSPACE-complete in this case. A further possibility is to syntactically restrict role value maps. For example in BACK [von Luck et. al, 1987], the lists of roles in role value maps are restricted to be lists of one element. Another possibility is to permit only role value maps in a role-defining style, i.e., only the form $(R) = (R_1,\ldots,R_n)$ is admissable, and there are no double definitions and no cycles. I suspect that subsumption in $\mathcal{ALR}$ becomes decidable under these restrictions.

Another direction of research is to use another semantics for the used constructs. This is a change of the meaning of the language and hence care should be taken. Nevertheless it may very well be the case that for small concepts the standard semantics fits our intuition, whereas for complex concepts, there may be a choice. Of course, the undecidability result depends on the imposed semantics of complex concepts, i.e., on the asymptotic behaviour of the semantics for large concepts.

Decidability of subsumption of feature terms can be interpreted as a change in the semantics of $\mathcal{ALR}$ or $\mathcal{ALRC}$, respectively. The approach to change the usual first-order logic to a four-valued as in [Patel-Schneider, 1987a, Patel-Schneider, 1987b] may help in re-establishing decidability to the price that the meaning of subsumption has changed. Corollary 3.7 suggests to prevent the formation of concepts that are consistent only in infinite models or to change the semantics such that only finite models are to be considered instead of all including infinite models. This idea is a remedy to Corollary 3.6 since consistency of concepts becomes recursively enumerable with respect to this semantics, but now the status of inconsistency of concepts or constraint systems is unclear and can be even worse than before.

A practical useful method is to put subsumption-problems in constraints and use constraint-propagation. The view taken here is that a user is not really interested in the subsumption relation of some complex concepts or in the consistency of concept, which he(she) has typed in, rather than in the answer to high-level queries that have as subproblems such subsumption or consistency tests. The idea is that the system computes as much as possible or as much as the user wants and then gives as answer a system C of constraints. These answer can be interpreted as follows: All solutions to C are solution to the query, or if only a yes/no answer is expected: If C has a solution, then the answer is yes, otherwise the answer is no. Such an approach is proposed in [Höhfeld and Smolka, 1988, Bläsius and Hedtstück, 1988, Jaffar and Lassez, 1986].

Acknowledgement

I would like to thank Jochen Doerre and Gert Smolka for discussion that contributed to the paper.

References

[Aït-Kaci, 1984] Hassan Aït-Kaci, A lattice theoretic approach to computation based on a calculus of partially ordered type structures, PhD Dissertation, Univ. of Pennsylvania, 1984

[Aït-Kaci and Nasr, 1986] Hassan Aït-Kaci, Roger Nasr, LOGIN: A Logic Programming Language with built-in inheritance. *J. Logic Programming* 3: 185-215, (1986)

[Bläsius and Hedtstück, 1988] Karl-Hans Bläsius, Ulrich Hedtstück, Resolution with feature unification, *Lecture Notes in Computer Science* 329: 17-26, (1988)

[Boone, 1959], W.W. Boone, The word problem, *Ann. Math.* (2), 70, 207-265, (1959)

[Brachman and Levesque, 1984] Ronald J. Brachman, Hector J. Levesque, The tractability of subsumption in frame-based description languages, *Proceedings AAAI-84*, pages 34-37, Austin, TX, 1984

[Brachman and Schmolze, 1985] Ronald J. Brachman, James G. Schmolze, An overview of the KL-ONE knowledge representation system. *Cognitive Science* 9(2): 171-216, 1985

[Brachman et. al., 1983] Ronald J. Brachman, Richard E. Fikes, Hector J. Levesque, Krypton: Integrating Terminology and Assertion, *Proceedings AAAI-83*, pages 31-35, Washington DC, 1983

[Brachman et. al., 1985] Ronald J.Brachman, Victoria P. Gilbert, Hector J. Levesque, An essential hybrid reasoning system: Knowledge and symbol level accounts of KRYPTON, Proceedings of the Ninth International Joint Conference on Artificial Intelligence 1985, pages 532-539, Los Angeles, 1985

[Burris and Sankappanavar, 1981] Stanley Burris, H.P. Sankappanavar, A course in universal algebra, Springer-Verlag, 1981

[Grätzer, 1979] George Grätzer, Universal Algebra, Springer-Verlag, 1979

[Höhfeld and Smolka, 1988], Markus Höhfeld, Gert Smolka, Definite relations over constraint languages, LILOG report 53, IBM Deutschland, West Germany, 1988

[Jaffar and Lassez, 1986], J. Jaffar, J.-L. Lassez, Constraint Logic Programming, 4th IEEE Symposium on Logic Programming, San Francisco, 1986

[Kaczmarek et. al. 1986] Thomas S. Kaczmarek, Raymond Bates, Gabriel Robins, Recent developments in NIKL, Proceedings AAAI-86, pages 978-985, 1986

[Kay, 1985], M. Kay, Parsing in functional unification grammar, In D. Dowty, L. Kartunnen and A. Zwicky (Eds.), Natural langauge parsing, Cambridge Universitiy Press, 1985

[Levesque and Brachman, 1985] Hector J. Levesque, Ronald J. Brachman, A fundamental tradeoff in knowledge representation, In *Readings in Knowledge Representation*, Morgan Kaufmann, 1985

[Levesque and Brachman, 1987] Hector J. Levesque, Ronald J. Brachman, Expressiveness and tractability in knowledge representation and reasoning, *Comput. Intell.* 3, 78-93, (1987)

[von Luck et. al., 1987] Kai von Luck, Bernhard Nebel, Christof Peltason, Albrecht Schmiedel, The anatomy of the BACK system. KIT-report 41, Technische Universität Berlin, F.R. Germany, 1987

[Mac Gregor and Bates, 1987] Robert M. Mac Gregor, Raymond Bates, The Loom knowledge representation language, Technical report ISI/RS-87-188, Information Sciences Institute, Univ. of Southern California, 1987

[Mac Gregor, 1987] Robert M. Mac Gregor, A deductive pattern matcher, Proceedings AAAI-88, (1988)

[Nebel, 1988], Bernhard Nebel, Computational complexity of terminological reasoning in BACK, *Artifical Intelligence* 34, 371-383, (1988)

[Novikov, 1955] P.S.Novikov,On the algorithmic undecidability of the word problem in group theory, *Trudy Mat. Inst. Steklov.* 14, Izdat. Nauk SSSR, Moscow, 1955

[Patel-Schneider, 1984] Peter F. Patel-Schneider, Small can be beautiful in knowledge representation, Proceedings *IEEE workshop on principles of*

knowledge based systems, pages 11-16, Denver, Colorado, 1984

[Patel-Schneider, 1989] Peter F. Patel-Schneider, Undecidability of subsumption in NIKL, (to appear in *Artificial Intelligence*), 1989

[Patel-Schneider, 1987a] Peter F. Patel-Schneider, Decidable, logic-based knowledge representation, PhD thesis, Dept. of Comp. Science, Univ. of Toronto, 1987
also: Technical report 56, Schlumberger Palo Alto Research, 1987,
also: Tech. report 201/87, Dept. of Comp. Science, Univ. of Toronto, 1987

[Patel-Schneider, 1987b] Peter F. Patel-Schneider, A decidable first-order logic for knowledge representation, draft, submitted to JAR, 1987

[Schild, 1988] Klaus Schild, Undecidability of subsumption in $\mathcal{U}$, draft, Institut für Software und theoretische Informatik, Technische Universität Berlin, 1988

[Schmidt-Schauß and Smolka, 1988] Manfred Schmidt-Schauß, Gert Smolka, Attributive concept descriptions with unions and complements, SEKI-report SR-88-21, Universität Kaiserslautern, 1988

[Schmolze and Israel, 1983], J.G. Schmolze, D.J. Israel, KL-ONE: Semantics and Classification. in: C.L. Sidner, Research in knowledge representation and natural language understanding, BBN Technical report 5421, Pages 27-39, Bolt, Beranek and Newman, Cambridge, MA, 1983

[Shieber, 1986] Stuart M. Shieber, An introduction to unification-based approaches to grammar, CSLI Lecture Notes 4, Stanford University, 1986

[Smolka and Aït-Kaci, 1987] Gert Smolka, Hassan Aït-Kaci, Inheritance hierarchies: Semantics and Unification. MCC Report AI-057-87, MCC, Austin, Texas, 1987

[Smolka, 1988], Gert Smolka, A feature logic with subsorts, LILOG report 33, IBM Deutschland, Stuttgart, West Germany, 1988

[Stillwell, 1982], J. Stillwell, The word problem and the isomorphism problem for groups, Bulletin AMS 6(1):33-56, 1982

[Vilain, 1985], Marc Vilain, The restricted language architecture of a hybrid representation system, Proceedings of the *Ninth International Joint Conference on Artificial Intelligence* 1985, pages 547-551, Los Angeles, 1985

TERMINOLOGICAL KNOWLEDGE REPRESENTATION SYSTEMS SUPPORTING N-ARY TERMS

James G. Schmolze
Department Of Computer Science
Tufts University
Medford, MA 02155 USA
Internet: schmolze@cs.tufts.edu

Abstract

We propose a new family of terminological knowledge representation (KR) systems that break the shackles of the one and two place relations. This family allows the direct representation of relations whose arity exceeds two while incorporating the advantages of current KR systems. We examine KR systems that restrict arity to two or less and find that, when representing relations of arity>2, users must go outside the KR system and are encouraged to use more awkward representation schemes. After examining the historical motivations for this arity≤2 restriction, we find that the advantages of allowing direct representation of n-ary relations far outweigh the reasons for the restriction. We sketch a family of n-ary KR systems and examine one member in detail. The exemplar system is an extension of the KANDOR system [Patel-Schneider, 1984] that we call NARY[KANDOR].

1 Introduction

Semantic Nets, *Frames* and the more recent terminological knowledge representation (KR) systems have been, and continue to be, of interest in Artificial Intelligence (AI). All of these systems provide devices for representing one and two place relations, and individuals. Semantic net systems such as that of Quillian [Quillian, 1967] provide *Generic Nodes* for one place relations, *Links* for two place relations, and *Individual Nodes* for individuals. Frame systems such as FRL [Roberts and Goldstein, 1977] provide *Generic Frames*, *Slots* and *Individual Frames*. More recent terminological KR systems such as KL-ONE [Brachman and Schmolze, 1985], KRYPTON [Brachman *et al.*, 1983], NIKL [Moser, 1983, Vilain, 1985], KANDOR [Patel-Schneider, 1984] and BACK [Luck *et al.*, 1987] provide *Concepts*, *Roles* and *Individuals*. In addition, two new KR systems now under development, CLASSIC [1] and LOOM [Mac Gregor, 1988] provide similar facilities. None of these systems provide facilities for directly representing three or more place relations.

Yet some relations are naturally expressed with arity greater than two. The *give* relation relates a giver, a receiver and an object given. The *between* relation relates two objects to a third that is located inbetween them. The *rent* relation connects a landlord, tenant, property, remuneration and, possibly, a period of time. Also, there is a large class of two and more place relations that, when time is taken into account, have their arity increased by one. For example, a time-sensitive *location* relation relates an object to its location at a given time. A time-sensitive *part-of* relation connects a part to its whole at a given time. Of course, arguments can be made that time should be treated specially, in which case we stand with our earlier examples. Finally, there are the ternary arithmetic relations such as *plus* or *times*.

The above KR systems present some problems to the user who must represent relations that have arity greater than two. The usual approach is to convert an n-ary relation to a unary relation plus n binary relations. For example, the *give* relation can be represented by a unary relation named, say, *give-occurrence* that represents giving occurrences, and a binary relation for each argument of *give*, say, *give-giver*, *give-receiver* and *give-object*. While this approach may suffice, its effective implementation usually requires additional reasoning mechanisms that are not provided by the above KR systems, and which therefore the user must provide. We examine these further in the next section.

Our approach is, initially, to make simple extensions to the above KR systems such that n-ary relations can be represented directly. We demonstrate this by extending the KANDOR system [Patel-Schneider, 1984] in Section 3. The selection of extensions has serious impact on the complexity of determining subsumption, a crucial algorithm for the more recent KR systems. The impact of language design on the complexity of

[1] Personal communication with Ronald J. Brachman, August 1988.

subsumption is discussed in general in [Brachman and Levesque, 1984]. In Section 4, we discuss its impact with respect to our work and, in Section 5, we present the central component of the subsumption algorithm. In Section 6, we sketch other n-ary KR languages. Finally, in Section 7 we draw conclusions about n-ary KR systems.

2 Why N-ary?

We continue with the *give* example except we add the time of the giving event as a fourth argument. Let *give(giver,receiver,object,time)* represent the giving of *object* from *giver* to *receiver* during time interval *time*. We represent *give* using only one and two place relations, which is done by introducing five new relations as explained earlier.

- *give-occurrence(occurrence)*: each element of *give-occurrence* corresponds to exactly one tuple of *give* and vice versa.
- *give-giver(occurrence,giver)*: relates each *give-occurrence* to its giver.
- *give-receiver(occurrence,receiver)*: relates each *give-occurrence* to its receiver.
- *give-object(occurrence,object)*: relates each *give-occurrence* to its object.
- *give-time(occurrence,time)*: relates each *give-occurrence* to the time interval during which it occurred.

Thus, *give(Mary,Joe,Ball,T)* is represented as

$$give\text{-}occurrence(GO) \wedge give\text{-}giver(GO,Mary) \wedge give\text{-}receiver(GO,Joe) \wedge give\text{-}object(GO,Ball) \wedge give\text{-}time(GO,T)$$

where *GO* is introduced to represent this particular giving occurrence.

However, we must also state that *give-giver*, *give-receiver*, *give-object* and *give-time* are total, single-valued functions.

$$\begin{aligned}
&\forall x[give\text{-}occurrence(x) \Rightarrow \exists y give\text{-}giver(x,y)] \wedge \\
&\forall x[give\text{-}occurrence(x) \Rightarrow \exists y give\text{-}receiver(x,y)] \wedge \\
&\forall x[give\text{-}occurrence(x) \Rightarrow \exists y give\text{-}object(x,y)] \wedge \\
&\forall x[give\text{-}occurrence(x) \Rightarrow \exists y give\text{-}time(x,y)] \wedge \\
&\forall x,y,z[give\text{-}giver(x,y) \wedge give\text{-}giver(x,z) \Rightarrow y = z] \wedge \\
&\forall x,y,z[give\text{-}receiver(x,y) \wedge give\text{-}receiver(x,z) \Rightarrow y = z] \wedge \\
&\forall x,y,z[give\text{-}object(x,y) \wedge give\text{-}object(x,z) \Rightarrow y = z] \wedge \\
&\forall x,y,z[give\text{-}time(x,y) \wedge give\text{-}time(x,z) \Rightarrow y = z]
\end{aligned} \tag{1}$$

Also, any two *give-occurrences* that have the same giver, receiver, object and time are identical.

$$\begin{aligned}
&\forall go1, go2, g, r, o, t \\
&\left(\begin{array}{l}
give\text{-}occurrence(go1) \wedge give\text{-}occurrence(go2) \wedge \\
give\text{-}giver(go1,g) \wedge give\text{-}giver(go2,g) \wedge \\
give\text{-}receiver(go1,r) \wedge give\text{-}receiver(go2,r) \wedge \\
give\text{-}object(go1,o) \wedge give\text{-}object(go2,o) \wedge \\
give\text{-}time(go1,t) \wedge give\text{-}time(go2,t)
\end{array}\right) \\
&\Rightarrow go1 = go2
\end{aligned} \tag{2}$$

Equation (2) is not strictly necessary but prevents clutter as we will see.

Equation (1) can be expressed in KANDOR and other similar systems, as we now show for *give* in KANDOR.[2]

```
(SLOT give-giver PRIMITIVE)
(SLOT give-receiver PRIMITIVE)
(SLOT give-object PRIMITIVE)
(SLOT give-time PRIMITIVE)
(FRAME give-occurrence PRIMITIVE
  (EXISTS give-giver 1)
  (EXISTS give-receiver 1)
  (EXISTS give-object 1)
  (EXISTS give-time 1)
  (ALL give-giver 1)
  (ALL give-receiver 1)
  (ALL give-object 1)
  (ALL give-time 1))
```

The semantics of the above is that *give-giver*, *give-receiver*, *give-object* and *give-time* each denote a primitive binary relation. *give-occurrence* denotes a primitive unary relation (i.e., a set) where each individual *give-occurrence* (call it *G*) satisfies the following necessary conditions.

1. *G* is *give-giver* related to exactly one other individual (the giver),
2. *G* is *give-receiver* related to exactly one other individual (the receiver),
3. *G* is *give-object* related to exactly one other individual (the object), and
4. *G* is *give-time* related to exactly one other individual (the time).

By examining the semantics of KANDOR (Figure 2), we see that, if the proper relations are associated with these primitive terms, equation (1) is a consequence of the above KANDOR specifications.

However, equation (2) cannot be expressed in KANDOR or other similar systems and leads to some difficulties when asserting information about individuals. Let us return to the earlier example of Mary giving a

[2] Names in formulas will be written in *italics* and names in term specifications will be written in a *slanted* font.

ball to Joe during time interval T. In KANDOR, this is represented with the following.[3]

(INDIVIDUAL *Mary*)
(INDIVIDUAL *Joe*)
(INDIVIDUAL *Ball*)
(INDIVIDUAL *T*)
(INDIVIDUAL *GO* (*give-occurrence*)
  ((*give-giver Mary*)
   (*give-receiver Joe*)
   (*give-object Ball*)
   (*give-time T*)))

We invented an individual *give-occurrence*, in this case *GO*, in order to assert the equivalent of *give(Mary,Joe,Ball,T)*. Unfortunately, with this approach there can be infinitely many representations of the same information. Namely, there could be any number of individual *give-occurrence* individuals that represent *give(Mary,Joe,Ball,T)*.

Users of these systems have two choices at this point. Either they can work with this potential redundancy, or they can eliminate it. To work with it, queries about the truth of the literal *give(Mary,Joe,Ball,T)* require search since this information may be associated with *any* individual *give-occurrence*. In other words, to determine if *give(Mary,Joe,Ball,T)*, one must do the following.

(for x in all individual *give-occurrence*'s
  is there an x such that
  (and (*give-giver* x *Mary*)
       (*give-receiver* x *Joe*)
       (*give-object* x *Ball*)
       (*give-time* x *T*)))

This is particularly frustrating in the KL-TWO system [Vilain, 1985], which uses the RUP [McAllester, 1982] system for representing assertions about individuals. In RUP, each literal is stored uniquely, which is implemented using an efficient hashing scheme. However, to identify the truth of *give(Mary,Joe,Ball,T)*, one cannot take full advantage of this hashing scheme since the individual *give-occurrence* is not known, and so one must revert to a more costly search.

If a user does not want this informational clutter, similar additional work must be done, but at the time of assertion instead of retrieval. Namely, if assertions about two distinct *give-occurrence*'s determine that they are equivalent, then the two individuals should be merged or equated.

In any case, the following is clear.

- There are many relations whose natural arity exceeds two.
- It is possible to represent relations of arity three or more in KANDOR and similar systems.
- However, one must go beyond the normal mechanisms of these systems to successfully do so.
- This extra work is eliminated if n-ary relations can be represented directly.

In addition, we will show the following.

- The representation of three or more place relations is much clearer and simpler in an n-ary system than in a KANDOR-like system.
- The complexity of determining subsumption need not be increased, but at the cost of limiting expressivity to that of KANDOR. Instead, we argue that increased expressivity is more important that having a complete, tractable subsumption algorithm.

This raises the question: Why haven't researchers designed n-ary terminological KR systems? While this question cannot be fully addressed here, one can trace the origins of the arity$\leq$2 restriction back to graphs. Graphs can only represent nodes and links, i.e., one and two place relations. Three place relations are usually depicted using a node and three links, i.e., a unary relation with three binary relations. Individuals are special types of nodes connected by special types of links. Graphs formed the basis of Semantics Networks, which in turn influenced Frame systems, which in turn influenced the more recent terminological systems. Throughout the development of these systems, the arity$\leq$2 restriction was maintained. The reasons for this are not clear, however, historical consistency played a part as did the fact that graphs were often used to depict the knowledge contained in these systems.

As these systems evolved, the more recent terminological systems such as KANDOR, KL-TWO and KRYPTON strengthened their connections to first order logic and weakened their connections to graphs. This, in turn, has weakened the arguments for retaining the arity$\leq$2 restriction, especially when going to an n-ary scheme involves only modest changes.

The remainder of this paper addresses the following research questions:

- *Can we design n-ary KR systems that are simple extensions of current KR systems?*
- *If so, how?*
- *If so, what does it cost?*

In this paper, we demonstrate that we can design such a system by extending KANDOR.

3 N-ary KANDOR

We will lay out one member of a family of n-ary KR systems. The family is called NARY; the member is

[3] In order to keep the example simple, we took a small liberty with KANDOR syntax and did not specify the types of *Mary*, *Joe*, *Ball* and *T*.

```
<definition>  ::=  <slot> |<frame> |<individual> |<decomp>

      <slot>  ::=  (SLOT <name>[<superslotname>] [<type>])
      <type>  ::=  <vr>

     <frame>  ::=  (FRAME <name> <superframename>* <prim> <restr>*)
      <prim>  ::=  DEFINED |PRIMITIVE
      <restr>  ::=  <somerestr> |<maxrestr> |<allrestr>
  <somerestr>  ::=  (EXISTS <slotname> [<vr>] [<minimum>])
  <maxrestr>  ::=  (ALL <slotname> <maximum>)
  <allrestr>  ::=  (ALL <slotname> <vr>)
         <vr>  ::=  <genvr> |<indvr>
      <genvr>  ::=  (GENERIC <framename>)
      <indvr>  ::=  (VALUE <val>)
        <val>  ::=  <individualname> |<number> |<string>
  <minimum>  ::=  "a number > 0"
  <maximum>  ::=  "a number ≥ 0"

 <individual>  ::=  (INDIVIDUAL <name> (<framename>+) <slotfiller>*)
   <slotfiller>  ::=  (<slotname> <val>+)

     <decomp>  ::=  (<name> <kind> <decomposedname> DISJOINT <elementname>+)
       <kind>  ::=  SLOT |FRAME

  <...name>  ::=  "a symbol"
```

A KANDOR specification is a sequence of <definition>'s. Items enclosed in "[ ]" are optional. "*" indicates zero or more repetitions. "+" indicates one or more repetitions. Use of double quotes indicates a general, non-syntactic, restriction. <number> and <string> have the obvious meaning. Frames should not be defined recursively. This specification is taken from [Patel-Schneider, 1984].

Figure 1: Syntax of KANDOR Specifications

called NARY[KANDOR]. Figure 1 shows the term specification language for KANDOR and Figure 2 gives its semantics as presented in [Patel-Schneider, 1984]. KANDOR has a strong reliance on the arity≤2 restriction. For example, the optional <type> specification for slots implicitly restricts the type of the second argument, i.e., it restricts the range. No type restriction for the first argument is allowed. In the <allrestr> for frames, the use of a <slotname> implies that the first argument of the slot refers to individuals of the given frame and the second argument refers to other individuals that must match the given <vr>.

In NARY systems, we allow for the specification of n-ary terms, where each term denotes an n-ary relation. One and two place terms are no longer special as they are in KANDOR. The syntax of NARY[KANDOR] is shown in Figure 3 and its semantics is shown in Figure 4.

We have renamed the EXISTS clause from KANDOR to MIN and have split the ALL clause of KANDOR into two different types of clauses: MAX and ALL. MIN specifies that a minimum number of tuples must exist that meet the given requirements. Similarly, MAX specifies a maximum number of tuples that can exist. ALL places type restrictions on elements of tuples related to individuals of the type being specified. As a result, the MAX restriction in NARY[KANDOR] can specify both a <maximum> and <vr> while the equivalent restriction in KANDOR can specify only a <maximum>, but not a <vr>.

Each term specification in NARY[KANDOR] includes a name for the term, a list of the formal parameters, a list of subsumers, an indication of either PRIMITIVE or DEFINED, and a list of restrictions.

- Each subsumer must have the same arity as the term being specified (similar to KANDOR but extended to n-ary).
- The indicator identifies the term as either PRIMI-

In a KANDOR knowledge base, let $\mathcal{F}$ be the set of frames defined in it, let $\mathcal{S}$ be the set of slots defined in it, let $\mathcal{R}$ be the set of restrictions used in frames in it, and let $\mathcal{I}$ be the set of individuals defined in it. A partial model for a KANDOR knowledge base is a set $\mathcal{D}$, the set of all individuals, plus a function, ξ, such that:

- $\xi : \mathcal{F} \longrightarrow 2^{\mathcal{D}}$
- $\xi : \mathcal{S} \longrightarrow (\mathcal{D} \rightarrow 2^{\mathcal{D}^+})$ where $\mathcal{D}^+$ is the disjoint union of D, integers and strings
- $\xi : \mathcal{R} \longrightarrow 2^{\mathcal{D}}$
- $\xi : \mathcal{I} \longrightarrow \mathcal{D}$ (an injective mapping)

where ξ must satisfy the following conditions:

1. for slots, s,
   (a) $\xi[s](x) \subseteq \xi[t](x)$ if s is defined with superslot t
   (b) $\xi[s](x) \subseteq \xi[f]$ if s is defined with type f
2. for restrictions
   (a) $\xi[s : \text{all } f] = \{x \in \mathcal{D} | \text{if } y \in \xi[s](x) \text{then } y \in \xi[f]\}$
   (b) $\xi[s : \text{all } i] = \{x \in \mathcal{D} | \text{if } y \in \xi[s](x) \text{then } y = \xi[i]\}$
   (c) $\xi[s :\leq n] = \{x \in \mathcal{D} | \; ||\xi[s](x)|| \; \leq n\}$
   (d) $\xi[s :\geq n \; f] = \{x \in \mathcal{D} | \; ||\xi[s](x) \cap \xi[f]|| \; \geq n\}$
   (e) $\xi[s :\geq n \; i] = \{x \in \mathcal{D} | \; ||\xi[s](x) \cap \{\xi[i]\}|| \; \geq n\}$
3. for frames, f,
   (a) $\xi[f] = \bigcap_{i=1}^{n} \xi[f_i] \; \cap \; \bigcap_{i=1}^{m} \xi[r_i]$ if f is defined as $f_1 \ldots f_n$ DEFINED $r_1 \ldots r_n$
   (b) $\xi[f] \subseteq \bigcap_{i=1}^{n} \xi[f_i] \; \cap \; \bigcap_{i=1}^{m} \xi[r_i]$ if f is defined as $f_1 \ldots f_n$ PRIMITIVE $r_1 \ldots r_n$
   (c) $\xi[f] \cap \xi[g] = \emptyset$ if f and g are members of the same disjoint decomposition.

$||S||$ denotes the cardinality of set S. This model is partial because it does not account for the definitions of individuals. This specification is taken from [Patel-Schneider, 1984].

Figure 2: Semantics of KANDOR Specifications

TIVE or DEFINED. As with KANDOR, the specification of a PRIMITIVE term introduces conditions that are necessary. The specification of a DEFINED term introduces conditions that are both necessary and sufficient.

- Restrictions are considerably more general than in KANDOR. Any formal parameter can be restricted by a TYPE, ALL, MIN or MAX clause. When restricting other relations using ALL, MIN or MAX clauses, any argument position can have its type restricted with a <vr>. (In KANDOR, type restrictions are done implicitly via subsumers for frames and can only be done for the second parameter of slots. Also, EXISTS and ALL restrictions can only appear on frames (i.e., unary terms) and implicitly restrict the first (i.e., only) parameter and the second argument of the given slot.)

As a result, the syntax of NARY systems requires that TYPE, ALL, MIN and MAX clauses identify the parameter and argument being restricted. In addition, when individuals are specified, the tuples that include the given individual can have the individual appear in any argument position (KANDOR requires that the individual appear in the first argument position).

In NARY[KANDOR] a term p *subsumes* a term q if and only if $\xi[q] \subseteq \xi[p]$ in all models of the given specifications. A second important relation is that of *realization*, where an individual *realizes* a unary term if it must be an instance of that term's denotation. Realization is a operation that KANDOR and similar systems perform automatically. In NARY[KANDOR], an individual i realizes a term p if and only if $\xi[i] \in \xi[p]$ in all models of the given specifications.

The following are some examples of NARY[KANDOR]. We first show the *give* example as a tertiary term.

<specification>	::=	<term> |<individual> |<decomp>
<term>	::=	(TOP <arity>) | (TERM [<name>] (<paramname>$^+$) <supertermorname>* <prim> <restr>*)
<prim>	::=	DEFINED |PRIMITIVE
<restr>	::=	<typerestr> |<allrestr> |<minrestr> |<maxrestr>
<typerestr>	::=	(TYPE <paramname> <unarytermorname>$^+$)
<allrestr>	::=	(ALL <paramname> <pluralliteral> <vr>)
<minrestr>	::=	(MIN <paramname> <pluralliteral> <minimum> [<vr>])
<maxrestr>	::=	(MAX <paramname> <pluralliteral> <maximum> [<vr>])
<vr>	::=	<genvr> |<indvr>
<genvr>	::=	<unaryliteral>
<indvr>	::=	(= <varname> <val>)
<val>	::=	<individualname> |<number> |<string>
<unaryliteral>	::=	(<unarytermorname> <paramorvarname>)
<pluralliteral>	::=	(<pluraltermorname> <paramorvarname>$^{++}$)
<paramorvarname>	::=	<paramname> |<varname>
<arity>	::=	"an integer > 0"
<minimum>	::=	"an integer > 0"
<maximum>	::=	"an integer $\geq$ 0"
<individual>	::=	(INDIVIDUAL <name> <unarytermorname>* <tuple>*)
<tuple>	::=	(<termorname> <val>*)
<decomp>	::=	(<name> <decomposedtermorname> DISJOINT <termorname>$^+$)
<...termorname>	::=	"either a <term> or the name of a term"
<...name>	::=	"a symbol"

Additional restrictions:

1. The <name> of a <term> must be supplied for PRIMITIVE terms.
2. The list of formal parameters for a <term> cannot contain duplicates. Their scope is the body of the term's specification.
3. Each <paramname> in a restriction must be a formal parameter of the term's specification.
4. In <allrestr>'s, <minrestr>'s and <maxrestr>'s,
   (a) the <paramname> must appear exactly once in the <pluralliteral>,
   (b) all other arguments to <pluralliteral> must be variables that are not formal parameters (their scope is limited to the body of the given restriction),
   (c) the variable used in the <vr> must be one of the variables used in the <pluralliteral> that is not a formal parameter, and
   (d) if <vr> is left out, it defaults to ((TOP 1) v) where v is any of the <varname>'s (i.e., not a <paramname>) in the <pluralliteral>.
5. In an <individual> specification, the individual's name must appear at least once in each <tuple>.

An NARY[KANDOR] specification is a sequence of <specification>'s where no term is specified more than once. The syntax is the same as in Figure 1. Terms may not be defined recursively. "++" indicates two or more repetitions.

Figure 3: Syntax of NARY[KANDOR] Specifications

In an NARY[KANDOR] knowledge base, let $\mathcal{T}$ be the set of terms defined, let $\mathcal{R}$ be the set of restrictions used, and let $\mathcal{I}$ be the set of individuals defined. A partial model for an NARY[KANDOR] knowledge base is a set $\mathcal{D}$, the set of all individuals, plus a function, ξ, such that:

- $\xi : \mathcal{T} \longrightarrow 2^{\mathcal{D}^*}$ where $\mathcal{D}^* = \mathcal{D}^1 \cup \ldots \cup \mathcal{D}^N$ and N is the highest arity of any term being specified.
- $\xi : \mathcal{R} \longrightarrow 2^{\mathcal{D}^*}$
- $\xi : \mathcal{I} \longrightarrow \mathcal{D}$ where this mapping is injective.

where ξ must satisfy the following conditions:

1. for restrictions:
   (a) $\xi[(\text{TYPE } p_j\ t_1 \ldots t_m)] = d_1 \times \ldots \times d_n$
   where p_j is the j-th parameter of an n-ary term, $d_j = \xi[t_1] \cap \ldots \cap \xi[t_m]$ and all other $d_i = \mathcal{D}$.

   (b) $\xi[(\text{ALL } p_j\ (f\ v_1 \ldots v_m)\ (t\ v_l))] = \{x \in \mathcal{D}^n | \forall y \text{ if } y \in \xi[f] \wedge x_j = y_k \text{ then } y_l \in \xi[t]\}$
   where p_j is the j-th parameter of an n-ary term, f is the name of the term being restricted whose arity is m, $p_j = v_k$ for some unique k, $1 \leq k \leq m$, all other v_i, $1 \leq i \leq m$, are distinct variables that are not parameters of the term, $1 \leq l \leq m$, and $v_l \neq p_j$.

   (c) $\xi[(\text{ALL } p_j\ (f\ v_1 \ldots v_m)\ (=\ v_l\ z))] = \{x \in \mathcal{D}^n | \forall y \text{ if } y \in \xi[f] \wedge x_j = y_k \text{ then } y_l = \xi[z]\}$
   where n, j, p_j, m, k, l and v_i, $1 \leq i \leq m$, are defined as above. This covers the case where the <vr> is <indvr>.

   (d) $\xi[(\text{MIN } p_j\ (f\ v_1 \ldots v_m)\ w\ (t\ v_l))] = \{x \in \mathcal{D}^n |\ ||\{y \in \xi[f] | x_j = y_k \wedge y_l \in \xi[t]\}|| \geq w\}$
   where n, j, p_j, m, k and v_i, $1 \leq i \leq m$, are defined as above and w is an integer with $w > 0$.

   (e) $\xi[(\text{MIN } p_j\ (f\ v_1 \ldots v_m)\ w\ (=\ v_l\ z))] = \{x \in \mathcal{D}^n |\ ||\{y \in \xi[f] | x_j = y_k \wedge y_l = \xi[z]\}|| \geq w\}$
   where n, j, p_j, m, k, w and v_i, $1 \leq i \leq m$, are defined as above. This covers the case where the <vr> is <indvr>.

   (f) $\xi[(\text{MAX } p_j\ (f\ v_1 \ldots v_m)\ w\ (t\ v_l))] = \{x \in \mathcal{D}^n |\ ||\{y \in \xi[f] | x_j = y_k \wedge y_l \in \xi[t]\}|| \leq w\}$
   where n, j, p_j, m, k and v_i, $1 \leq i \leq m$, are defined as above and w is an integer with $w \geq 0$.

   (g) $\xi[(\text{MAX } p_j\ (f\ v_1 \ldots v_m)\ w\ (=\ v_l\ z))] = \{x \in \mathcal{D}^n |\ ||\{y \in \xi[f] | x_j = y_k \wedge y_l = \xi[z]\}|| \leq w\}$
   where n, j, p_j, m, k, w and v_i, $1 \leq i \leq m$, are defined as above. This covers the case where the <vr> is <indvr>.
2. for terms:
   (a) $\xi[(\text{TOP } n)] = \mathcal{D}^n$

   (b) $\xi[(\text{TERM } t\ (p_1 \ldots p_n)\ s_1 \ldots s_k \text{ DEFINED } r_1 \ldots r_m)] = \xi[t] = \mathcal{D}^n \cap \xi[s_1] \cap \ldots \cap \xi[s_k] \cap \xi[r_1] \cap \ldots \cap \xi[r_m]$

   (c) $\xi[(\text{TERM } t\ (p_1 \ldots p_n)\ s_1 \ldots s_k \text{ PRIMITIVE } r_1 \ldots r_m)] =$
   $\xi[t] \subseteq \mathcal{D}^n \cap \xi[s_1] \cap \ldots \cap \xi[s_k] \cap \xi[r_1] \cap \ldots \cap \xi[r_m]$

   (d) $\xi[f] \cap \xi[g] = \emptyset$
   if f and g are elements of the same disjoint decomposition.

$||S||$ denotes the cardinality of set S. This model is partial because it does not account for the specifications of individuals.

Fi urg 4: Semantics of NARY[KANDOR] Specifications

```
(TERM give (giver receiver object time)
  PRIMITIVE)
```

For comparison, we now repeat the specification of *give* from Section 2 using the unary/binary format that KANDOR requires.

```
(SLOT give-giver PRIMITIVE)
(SLOT give-receiver PRIMITIVE)
(SLOT give-object PRIMITIVE)
(SLOT give-time PRIMITIVE)
(FRAME give-occurrence PRIMITIVE
  (EXISTS give-giver 1)
  (EXISTS give-receiver 1)
  (EXISTS give-object 1)
  (EXISTS give-time 1)
  (ALL give-giver 1)
  (ALL give-receiver 1)
  (ALL give-object 1)
  (ALL give-time 1))
```

It is quite plain that the direct representation of *give* as a tertiary term is considerably clearer and simpler than as a unary relation with four binary relations. In addition, the tertiary representation eliminates the problems of informational clutter that can arise when assertions are made about individuals (as discussed in Section 2).

Using the tertiary *give* from above, we now define a giver of apple seeds, a giver of many apple seeds (i.e., ≥ 20), an individual apple seed giver, and a receiver of only apple seeds.

```
(TERM apple-seed-giver (asg) person DEFINED
  (MIN asg (give asg r as t) 1 (apple-seed as)))
(TERM many-apple-seed-giver (asg) person
  DEFINED
  (MIN asg (give asg r as t) 20 (apple-seed as)))
(INDIVIDUAL Johnny-apple-seed apple-seed-giver
  (give Johnny-apple-seed Mary Apple-Seed-37 T34)
  (give Johnny-apple-seed Joe Apple-Seed-82 T65))
(TERM receiver-of-only-apple-seeds (asr) person
  DEFINED
  (ALL asr (give g asr as t) (apple-seed as)))
```

Some variables, such as r and t in *apple-seed-giver*, are used merely as place holders.

Alternatively, we can define *apple-seed-giver* by using a restricted version of *give* that represents the giving of apple seeds.

```
(TERM give-apple-seed (g r o t) give DEFINED
  (TYPE o apple-seed))
(TERM apple-seed-giver (asg) person DEFINED
  (MIN asg (give-apple-seed asg r as t) 1))
```

As the above example demonstrates, NARY[KANDOR] allows for *vrdiff* specifications [Brachman and Levesque, 1984].

The following specifies occurrences of giving an apple to a parent.[4]

```
(TERM give-apple-to-parent (g r o t) give
  DEFINED
  (TYPE o apple)
  (MIN r (offspring r c) 1))
```

Next, we examine the (somewhat over-used) person, parent and grandparent example, which needs only unary and binary terms.

```
(TERM person (p) PRIMITIVE)
(TERM offspring (parent child) PRIMITIVE
  (TYPE parent person)
  (TYPE child person))
(TERM parent (p) person DEFINED
  (MIN p (offspring p c) 1))
(TERM grandparent (g) person DEFINED
  (MIN p (offspring p c) 1 (parent c)))
```

Finally, we contrast KANDOR and NARY[KANDOR] by listing the additional restrictions one needs to apply to NARY[KANDOR] to reduce it to the expressive power of KANDOR.

- The arity of all terms must be one ("frames") or two ("slots").
- Only unary terms can be DEFINED. All others must be PRIMITIVE.
- The TYPE restriction can only apply to the second parameter of binary terms.
- ALL, MIN and MAX restrictions can only apply to unary terms.
- In an ALL, MIN or MAX restriction, the relation being restricted must be binary, must have its first argument apply to the parameter of the term being defined and must have its second argument further restricted.
- In a MAX specification, the <vr> must be left out.
- Each tuple in an INDIVIDUAL specification must have the given individual as the first argument.

Given the context of an n-ary system, some of the restrictions of KANDOR seem arbitrary. However,

[4] It turns out that we can restrict the use of ALL, MIN or MAX clauses to unary terms only, and still retain the same expressive power in NARY[KANDOR]. An equivalent specification can always be made using the TYPE restriction. For example, the following is an equivalent specification of *give-apple-to-parent*.

```
(TERM give-apple-to-parent (g r o t) give
  DEFINED
  (TYPE o apple)
  (TYPE r (TERM (x) DEFINED
            (MIN x (offspring x c) 1))))
```

However, we find the original syntax easier to read.

KANDOR was originally designed to have a tractable and complete subsumption algorithm. While this goal turns out to have failed [Nebel, 1988], it played an important role in the design of KANDOR.

4 Subsumption

In any KR system, there is a balance between the expressiveness of the language versus the complexity of drawing inferences. In terminological systems, this balance focuses on the expressivity of the language versus the complexity of subsumption.

KANDOR is moderately expressive yet subsumption is not tractable in it [Nebel, 1988]. NARY[KANDOR] is more expressive than KANDOR and hence subsumption is not tractable in it. In fact, subsumption is probably worse in NARY[KANDOR] because it allows for *vrdiff* constructs [Brachman and Levesque, 1984].

There are many ways to address this balance. KRYPTON [Brachman *et al.*, 1983] was designed to have tractable subsumption, which forced it to have a very weak language. Experience with KL-ONE [Brachman and Schmolze, 1985] and NIKL [Moser, 1983, Vilain, 1985] was quite different. First, there was always a sizable community of users for these two systems. Second, user's always wanted more expressive power, not less, and were very willing to accept incomplete subsumption algorithms as the cost of greater expressivity.

Our approach is similar to that of the designers of KL-ONE and NIKL. We prefer a language with useful expressive power even though subsumption is not tractable. The subsumption algorithm provided would be tractable (and fast, hopefully), but incomplete. To help with this incompleteness, we hope to identify restricted subsets of the language such that, if an application remains within that subset, the subsumption algorithm will be both tractable and complete. In other words, if we make these identifications, users who remain within the restricted subsets are rewarded with complete subsumption. If users go beyond that subset, they must deal with an incomplete yet tractable subsumption algorithm.

5 Subsumption Algorithm

We now sketch the subsumption algorithm for a subset of NARY[KANDOR]. A full presentation is not possible, so we outline the algorithm and focus on the central component that compares two n-ary terms.

When a term is specified to the system, a process is performed called *completion* (as is done in the NIKL system). Completion performs all inheritance operations along with a number of local inference operations to arrive at an expanded but equivalent form of the term. This expanded form replaces the original specification.

Let us assume we are completing a specification S. One way to perform inheritance is to replace all the names of terms by their respective specifications. For DEFINED terms, we do not include the term's name in the new specification. For PRIMITIVE terms, we retain the term's name (since the specification is incomplete, we use the term's name to represent the "rest" of the meaning of the term). Then a number of simple, local inference operations are performed. Finally, the specification is simplified and normalized.

The final form of a specification after completion is

$$(\text{TERM } t\ (p_1 \ldots p_n)\ s_1 \ldots s_k\ dp\ r_1 \ldots r_m)$$

where each s_i is the name of a primitive term, dp is either DEFINED or PRIMITIVE, and each r_i is a simplified, normalized restriction. The names of defined terms do not appear anywhere in the completed specification.

After completion, we are able to test for subsumption. Below, we present the algorithm that compares two completed specifications to determine whether one subsumes the other. Using this algorithm, the system can perform *classification*, which takes a newly specified term, completes it and compares it to all other terms in the knowledge base to discover and record any new subsumption relations.

The algorithm for structural subsumption, SS(T^A, T^B), makes a structural (i.e., syntactic) comparison between completed specifications T^A and T^B to determine whether T^A subsumes T^B. SS returns either **true** or **false**. **true** indicates that T^A subsumes T^B. However, since our algorithm is incomplete, **false** merely indicates that the system cannot show that T^A subsumes T^B (i.e., it is unknown). SS depends upon SSR(R^A, R^B), which structurally compares two restrictions to determine whether R^A subsumes R^B. Both algorithms are shown in Figure 5.

Interestingly, these algorithms are very similar to the algorithm used in KL-ONE [Schmolze and Israel, 1983, Schmolze and Lipkis, 1983] and its successor NIKL (and is likely similar to other subsumption algorithms). A few components of the original algorithm were generalized to incorporate the n-ary nature of NARY[KANDOR].

Our presentation of SS and SSR is abbreviated. We note that we have left out <indvr>, <decomp> and <individual>. We have not yet done an analysis of the algorithm's complexity. However, it is certainly polynomial in the size of the terms and appears to be, at worst, $O(n^3)$, where n is a measure of the size of the terms.

6 Other N-ary KR Languages

Many constraints of NARY[KANDOR] can be relaxed to produce a family of n-ary languages. The following is a list of possibilities.

$\mathbf{SS}(T^A, T^B) \stackrel{\text{def}}{=}$
if $T^A = T^B$ **then true**
else cond
case $T^A = (\text{TOP } n)$:
$T^B = (\text{TOP } n)$ **or** $T^B = (\text{TERM } [t]\ (p_1 \ldots p_n)\ s_1 \ldots s_k\ dp\ r_1 \ldots r_m)$
case $T^B = (\text{TOP } n)$: **false**
case $T^A = (\text{TERM } [t^A]\ (p_1^A \ldots p_n^A)\ s_1^A \ldots s_{k_A}^A\ \text{DEFINED}\ r_1^A \ldots r_{m_A}^A)$:
$T^B = (\text{TERM } [t^B]\ (p_1^B \ldots p_n^B)\ s_1^B \ldots s_{k_B}^B\ dp\ r_1^B \ldots r_{m_B}^B)$ **and** $\{s_1^A, \ldots, s_{k_A}^A\} \subseteq \{s_1^B, \ldots, s_{k_B}^B\}$ **and**
for each a, $1 \le a \le m^A$, there is a b, $1 \le b \le m^B$, such that $\text{SSR}(r_a^A, r_b^B)$
case $T^A = (\text{TERM } t^A\ (p_1^A \ldots p_n^A)\ s_1^A \ldots s_{k_A}^A\ \text{PRIMITIVE}\ r_1^A \ldots r_{m_A}^A)$:
$T^B = (\text{TERM } [t^B]\ (p_1^B \ldots p_n^B)\ s_1^B \ldots s_{k_B}^B\ dp\ r_1^B \ldots r_{m_B}^B)$ **and** $t^A \in \{s_1^B, \ldots, s_{k_B}^B\}$
end

$\mathbf{SSR}(R^A, R^B) \stackrel{\text{def}}{=}$
cond
case $R^A = (\text{TYPE } p_j^A\ t_1^A \ldots t_{m_A}^A)$:
$R^B = (\text{TYPE } p_j^B\ t_1^B \ldots t_{m_B}^B)$ **and**
for each a, $1 \le a \le m^A$, there is a b, $1 \le b \le m^B$, such that $\text{SS}(t_a^A, t_b^B)$
case $R^A = (\text{ALL } p_j^A\ (f^A\ v_1^A \ldots v_m^A)\ (t^A\ v_l^A))$:
$R^B = (\text{ALL } p_j^B\ (f^B\ v_1^B \ldots v_m^B)\ (t^B\ v_l^B))$ **and**
for some $k, 1 \le k \le m$, $(p_j^A = v_k^A$ **and** $p_j^B = v_k^B)$ **and**
$\text{SS}(f^B, f^A)$ **and** $\text{SS}(t^A, t^B)$
case $R^A = (\text{MIN } p_j^A\ (f^A\ v_1^A \ldots v_m^A)\ w^A\ (t^A\ v_l^A))$:
$R^B = (\text{MIN } p_j^B\ (f^B\ v_1^B \ldots v_m^B)\ w^B\ (t^B\ v_l^B))$ **and**
for some $k, 1 \le k \le m$, $(p_j^A = v_k^A$ **and** $p_j^B = v_k^B)$ **and**
$\text{SS}(f^A, f^B)$ **and** $w^A \le w^B$ **and** $\text{SS}(t^A, t^B)$
case $R^A = (\text{MAX } p_j^A\ (f^A\ v_1^A \ldots v_m^A)\ w^A\ (t^A\ v_l^A))$:
$R^B = (\text{MAX } p_j^B\ (f^B\ v_1^B \ldots v_m^B)\ w^B\ (t^B\ v_l^B))$ **and**
for some $k, 1 \le k \le m$, $(p_j^A = v_k^A$ **and** $p_j^B = v_k^B)$ **and**
$\text{SS}(f^B, f^A)$ **and** $w^A \ge w^B$ **and** $\text{SS}(t^B, t^A)$
end

Notation:

- The **cond** executes the *first* **case** whose test is **true**.
- Any item enclosed in [ ] is optional.
- SSR assumes that its arguments are from terms with the same arity. It also knows (via means not shown here) that each p_j is the j-th parameter of the corresponding term. For SSR to be true, the same parameter position must be restricted in each clause. Also, the <vr> of each clause must restrict the same argument position of the corresponding f's.
- For the ALL, MIN and MAX cases,
  - f is the name of the term being restricted and m is it's arity,
  - $p_j = v_k$ for some unique k, $1 \le k \le m$ (i.e., the parameter being restricted appears as the k-th argument of f for some k),
  - all other v_i, $1 \le i \le m$ where $i \ne k$ (i.e., all arguments of f except v_k), are distinct variables that are not parameters of the term being specified,
  - v_l, $1 \le l \le m$, is some variable appearing as the l-th argument to f where $v_l \ne p_j$ (i.e., v_l is not the parameter being restricted).

Figure 5: The Structural Subsumption Algorithm

- Allow multiple parameters to be restricted by non-unary terms in a TYPE restriction.
- Allow multiple <vr>'s in ALL, MIN and MAX restrictions.
- Allow other than unary terms in the <vr>'s of ALL, MIN and MAX restrictions.
- Allow several parameters of the term being restricted within a single ALL, MIN or MAX restriction, or allow one parameter to appear more than once.
- Allow for subset and/or equality restrictions similar to those found in KL-ONE and NIKL.
- Allow a richer representation of individuals. For example, do not require that $\xi : \mathcal{I} \longrightarrow \mathcal{D}$ be injective. Allow for a more expressive language for assertions about individuals, such as the propositional calculus that is part of the KL-TWO system [Vilain, 1985].
- Allow for recursive term specifications. See [Nebel, 1987] for a discussion of the semantics for such languages.

By relaxing some or all of these constraints, we can produce a family of n-ary KR languages. We hope to explore some of these in the future.

7 Conclusions

N-ary KR systems are possible and need not be complicated. In contrast to KANDOR-like systems, n-ary systems greatly simplify the representation of relations with arity>2 and eliminate the problems with tracking individuals in such relations.

We have sketched a family of n-ary systems and have presented one member, NARY[KANDOR], that extends the KANDOR system. The expressiveness of NARY[KANDOR] is considerably more than that of KANDOR, which is good for expressiveness and bad for the complexity of subsumption. However, we argue that this is a good overall design decision. This argument is based primarily on the KL-ONE and NIKL experience where users did not want weakly expressive systems, even though it meant that complete subsumption could not be computed. For those who are interested in designing a weaker n-ary system where subsumption has the same complexity as KANDOR, we identified the differences between the KANDOR and NARY[KANDOR] systems in terms of expressive power.

Overall, we argue that the KR community should embrace n-ary systems, with NARY[KANDOR] serving as an example of how it might be done.

Acknowledgements

Thanks goes to Tom Lipkis and Rusty Bobrow who, when implementing the NIKL classifier several years ago, observed that subsumption for Roles was very similar to subsumption for Concepts and suggested that an n-ary NIKL should not be too difficult to design. Also, thanks to David Israel, Ron Brachman, Brad Goodman and this paper's reviewer for comments on earlier drafts.

References

[Brachman and Levesque, 1984] Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of AAAI-84*, pages 34–37, Austin, Texas, August 1984. American Association for Artificial Intelligence.

[Brachman and Schmolze, 1985] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, April–June 1985.

[Brachman *et al.*, 1983] Ronald J. Brachman, Richard E. Fikes, and Hector J. Levesque. KRYPTON: A functional approach to knowledge representation. *IEEE Computer*, 16(10):67–73, October 1983. Also available as Fairchild Technical Report No. 639 or as FLAIR Technical Report No. 16.

[Luck *et al.*, 1987] K. von Luck, B. Nebel, C. Peltason, and A. Schmiedel. The anatomy of the BACK system. Technical Report KIT 47, Technische Universitat Berlin, Berlin, West Germany, 1987.

[Mac Gregor, 1988] Robert M. Mac Gregor. A deductive pattern matcher. In *Proceedings of AAAI-88*, pages 403–408, St. Paul, Minn., August 1988. American Association for Artificial Intelligence.

[McAllester, 1982] David A. McAllester. Reasoning utility package user's manual. AI Memo 667, Massachusetts Institute of Technology Artificial Intelligence Laboratory, April 1982.

[Moser, 1983] M. G. Moser. An overview of NIKL, the new implementation of KL-ONE. In *Research in Knowledge Representation for Natural Language Understanding - Annual Report, 1 September 1982 - 31 August 1983*, pages 7–26. BBN Laboratories Report No. 5421, 1983.

[Nebel, 1987] Bernhard Nebel. On terminological cycles. Technical Report KIT - Report 58, Technische Universitat Berlin, West Germany, November 1987.

[Nebel, 1988] Bernhard Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence Journal*, 34(3):371–383, April 1988.

[Patel-Schneider, 1984] Peter F. Patel-Schneider. Small can be beautiful in knowledge representation. In *Proceedings IEEE Workshop on Principles of Knowledge-Based Systems*, pages 559–565, Denver, Colorado, December

1984. IEEE Computer Society. Extended version available as AI Technical Report No. 37, Schlumberger Palo Alto Research, October 1984.

[Quillian, 1967] M. Ross Quillian. Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science*, 12(5), 1967.

[Roberts and Goldstein, 1977] R. Bruce Roberts and Ira P. Goldstein. *The FRL Manual.* Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, MA, 1977. MIT AI Lab Memo 409.

[Schmolze and Israel, 1983] James G. Schmolze and David J. Israel. KL-ONE: Semantics and classification. In *Research in Knowledge Representation for Natural Language Understanding, Annual Report (1 Sept. 1982 - 31 Aug. 1983). BBN Report No.@ 5421*, pages 27–39. Bolt Beranek and Newman Inc, Cambridge, MA, 1983.

[Schmolze and Lipkis, 1983] James G. Schmolze and Thomas A. Lipkis. Classification in the KL-ONE knowledge representation system. In *Proceedings of IJCAI-83.* International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, August 1983.

[Vilain, 1985] Marc Vilain. The restricted language architecture of a hybrid representation system. In *Proceedings IJCAI-85*, pages 547–551, Los Angeles, California, August 1985. International Joint Conference on Artificial Intelligence.

An Episodic Knowledge Representation for Narrative Texts

Lenhart K. Schubert & Chung Hee Hwang*

Department of Computer Science
University of Rochester
Rochester, New York 14627

Abstract

We would like to build story understanding systems which are transparent, modular, and extensible. To this end, we have been working on a new logical approach to narrative understanding that features a *GPSG-style grammar* and an *episodic logic* with *probabilistic* inference rules. The grammar represents phrase structure and the relationship between phrase structure and logical form in a modular, explicit form. The logical representation allows propositional attitudes, unreliable generalizations, and other non-standard constructs, providing a uniform, transparent knowledge representation for both the explicit content of stories and for the background knowledge needed to understand them. It makes systematic use of *episodic* variables in the representation of episodic sentences, using these to capture temporal and causal relationships. The rules of inference include probabilistic versions of natural deduction rules and rules resembling "rule instantiation" in expert systems. These can be used for predictive, explanatory, and simulative inference. We illustrate our approach with nontrivial grammar fragments (including semantic rules), and with an extended example of forward-chaining inference based on a sentence from "Little Red Riding Hood." A pilot implementation is able to make many (though not all) of the inferences we describe.

1 Introduction

Many ideas and systems have been developed for narrative understanding, and some of them, e.g., CYRUS (Kolodner 1981) and BORIS (Lehnert *et al.* 1983, Dyer 1983), have shown a remarkable degree of understanding in complicated human domains. These ambitious systems, however, are very complex and hard to extend beyond the few stories they handle. We believe this is so because of insufficient *expressiveness* and *clarity* of the knowledge representations used, for knowledge about language as well as about the world, and about inferences warranted by that knowledge.

For example, the meaning representations used often cannot express complex quantification ("most people with two or more cars"), logical compounding ("If he fails, he is either lazy or a fool"), complex concepts ("the type of person who never forgets a slight"), modification ("a nearly invisible pale brown birthmark"), temporal relations ("He had seen her twice the previous week"), and so on. As well, knowledge about language and about the world are often buried in procedures (e.g., procedures which seek semantically appropriate fillers for frame slots) in a way that makes it very hard to determine what linguistic and factual assumptions have been made.

These considerations have led us to an approach to narrative understanding in which all types of linguistic, world and inference knowledge are represented in an explicit, analyzable form. We have chosen Generalized Phrase Structure Grammar (GPSG) as our grammatical representation and have been developing a knowledge representation, called *Episodic Logic*, for encoding both the content of narratives and the knowledge needed to understand them. GPSG is a particularly perspicuous grammatical formalism which is expressively adequate for almost all English grammatical phenomena, and is relatively easy to use by a parser and logical-form generator. Episodic logic is expressively rich and close enough to surface form so that the relationship between surface form and logical form can be specified in a modular, transparent way. It introduces episodic variables so that implicit, context-dependent relationships among episodes (events, ac-

*Part of this research was done while the authors were at the Department of Computing Science, University of Alberta, Edmonton, Canada.

tions, situations, etc.) can be made explicit. It also allows the representation of restricted quantifiers, propositional attitudes, predicate modifiers, nominalized predicates, and perhaps most importantly, unreliable generalizations. Such generalizations have recently received much attention in the non-monotonic reasoning literature and elsewhere (e.g., linguistic semantics). The practical adequacy of our logic has been tested on small story fragments.

In the next section, we motivate and explain some of the unusual features of our logic. In section 3, we provide glimpses of semantics. Then in section 4, we sketch the derivation of episodic logical form from surface structure using a GPSG grammar. Next, in section 5, we introduce some inference rules and indicate their role within our implementation. In section 6, we illustrate the inference process in story understanding with an extended example based on a small fragment of *Little Red Riding Hood*. In the concluding section, we comment on related work and assess the progress made and work still to be done.

2 Episodic Logic

Our initial logical form for English sentences is quite close to surface structure; it "mimics" noun phrases in its use of restricted quantifiers, allows for predicate modification and λ-abstraction, and follows English surface syntax by having the "subject" of a predication precede the predicate. For example, "Every dog has a tail" would be represented as

(1) $(\forall x{:}[x \text{ dog}]\ (\exists y{:}[y \text{ tail}]\ [x \text{ have-as-part } y]))$[1]

after predicate disambiguation and quantifier scoping. (The initial representation computed from the surface form would be [<$\forall$ dog> have <$\exists$ tail>].)

Two more features that lead to close conformity between the surface form and our logical form are predicate modification and λ-abstraction, illustrated by (2) and (3):

(2) a. Canada is very distant from Australia
b. [Canada (very $(\lambda x[x$ distant-from Australia$]))$]

(3) a. the brother of Mary who is a doctor
b. <THE $(\lambda x[[x$ brother-of Mary$] \wedge [x$ doctor$]])$ >

Here, the predicate modifier *very* is a function which when applied to a predicate yields another, more restricted predicate. In (3), the term in **b** is the unscoped translation of the phrase in **a** .

Another distinctive feature of our logic, responsible for its name, is the inclusion of *episodic variables*. Whereas examples (1) - (3) dealt with "atemporal" or "nonepisodic" properties, (4) and (5) below deal with "episodic" ones, in the representation of which episodic variables play a key role.

(4) Everyone looked at Mary;
(5) This (event) made her blush.

Note that (5) refers deictically to the episode of *everyone looking at Mary*, and relates it causally to another episode, that of *Mary's blushing*. Using a connective "$**$" relating a sentence to the episode it characterizes, we can represent (4) and (5) as

(4') $(\exists e1{:}[e1$ before *now*$]$[2]
$[(\forall x{:}[x \text{ person}][x \text{ look-at Mary}]) ** e1])$
(5') $(\exists e2{:}[e2$ before *now*$]$
$[[e1 \text{ cause-of } e2] \wedge [[\text{Mary blush}] ** e2]])$.

Thus (4') says that $e1$ is an episode with characterization "Everyone looked at Mary," and similarly for $e2$ in (5'). A characterizing description is not necessarily a maximally specific one. That is, "Everyone looked pointedly at Mary," "Everyone looked at someone," or even "Some people did something" may all be characterizations of one and the same episode.

Our logic also contains a weaker but more fundamental operator "$*$" that reads "is a partial description of." "$*$" is essentially an object-language embedding of the semantic notion of *truth over an episode or situation*. For example, $[[\text{Mary blush}] * e]$ entails the truth of [Mary blush] over episode e. (The blushing must extend over the entire episode.) "$**$" is a special case of "$*$", holding only if the given partial description *characterizes* the given situation. To see the significance of the distinction between "characterizations" and arbitrary partial descriptions of episodes, suppose that the "$**$" in (4') were replaced by "$*$". Then (as a little thought shows) (4') and (5') would be true in a situation in which everyone looked at Mary, *laughing derisively*, and it was *this* (more complex) event that made her blush. Yet we would not say that (4) and (5) are both true in such a situation; so (4'),

[1] We use restricted quantification of the form $(Q\alpha{:}\Phi\Psi)$, where Q is a quantifier, α is a variable, and Φ and Ψ are formulas. That is, $(\forall\alpha{:}\Phi\Psi)$ and $(\exists\alpha{:}\Phi\Psi)$ are equivalent to $(\forall\alpha)\Phi \rightarrow \Psi$ and $(\exists\alpha)\Phi \wedge \Psi$, respectively. When there is no restriction Φ, we write $(Q\alpha\Psi)$. Also note that we use square brackets to indicate predicate infix expressions, round brackets for prefix expressions, and angle brackets for unscoped operators. Scoping of quantifiers and other operators is discussed in Schubert & Pelletier (1982), Hurum & Schubert (1986), and Hurum (1987,1988).

[2] Note the reduction of past tense to a relation placing episode $e1$ before *now* (an indexical constant to be immediately replaced by a term with fixed reference to the time of speech).

with "**" weakened to "*", would not be a correct formalization of (4).

Another point to be noted above is the *free* occurrence of variable $e1$ in (5'), outside the scope of its quantifier. This is permissible in our logic, thanks to a "context change" mechanism (Schubert & Pelletier 1988) which "carries forward bindings" of existential variables *out* of the scopes of the existential quantifiers, to subsequent clauses. As Heim(1982), Kamp(1981) and many others have observed, some such mechanism seems essential to deal with anaphora in extended discourse.

This mechanism is also the key to representing generic conditionals, such as "A wolf is (usually) grey," "A child (usually) loves his or her grandmother," or "When two strangers meet in a deserted region, they often greet." Generic conditionals take the following form:

$$(\exists x_1(\exists x_2(\cdots(\exists x_k \Phi)\cdots))) \rightarrow_p \Psi,$$

where Φ and Ψ are sentences involving $x_1, x_2, \ldots, x_k$, and p is a numeric lower bound on frequency (objective probability). Thus,

$$(\exists x[x \text{ wolf}]) \rightarrow_{.8} [x \text{ grey}]$$

says that a wolf is usually grey (or, most wolves are grey). Similarly, to express that when a predator encounters a smaller non-predatory animal, he may attack it, we would use a generalization involving an existentially quantified predator, non-predatory animal, and encounter episode in the antecedent. Generic conditionals are often used in *causal* axioms. In particular, *predictive* causal axioms assume the occurrence of some particular type of episode $e1$ in the antecedent, and predict another episode $e2$ caused by $e1$ in the consequent. The following is an example of a predictive axiom:

> *When a predatory animal sees a non-predatory creature of comparable or smaller size, it may want to attack and eat it.*

$$\begin{array}{l}(\exists x{:}[x \text{ predatory-animal}] \\ \quad (\exists y{:}[[y \text{ creature}] \wedge \neg[y \text{ predatory-animal}] \wedge \\ \qquad [[y \text{ as-big-as } x] \vee [y \text{ smaller-than } x]]] \\ \quad\ (\exists e1[[x \text{ see } y] ** e1])))\\ \rightarrow_{.6} [(\exists e2{:}[[(\text{begin } e2) \text{ during } e1] \wedge [e1 \text{ cause-of } e2]] \\ \qquad [[x \text{ want} \\ \qquad\ (\text{To } (\lambda e \lambda x'[[x' \text{ attack } y] ** e]))] ** e2]) \wedge \\ \quad (\exists e3{:}[[(\text{begin } e3) \text{ during } e1] \wedge [e1 \text{ cause-of } e3]] \\ \qquad [[x \text{ want} \\ \qquad\ (\text{To } (\lambda e \lambda x'[[x' \text{ eat } y] ** e]))] ** e3])]^3\end{array}$$

Equally important are *explanatory* axioms, such as the following:

> *If a creature wants to eat some food, it is likely to be hungry.*

$$\begin{array}{l}(\exists x(\exists e1[[x \text{ want} \\ \qquad (\text{To } (\lambda e \lambda x'(\exists y{:}[y \text{ food-for } x'] \\ \qquad\ [[x' \text{ eat } y] ** e]))))] ** e1])) \\ \rightarrow_{.9} (\exists e2{:}[[e2 \text{ cause-of } e1] \wedge [e2 \text{ same-time } e1]] \\ \qquad [[x \text{ hungry}] * e2])\end{array}$$

Much of the world knowledge we use in our experimentation is in fact stated as causal axioms like these. Probabilistic inference rules allow such axioms to be used to make more or less uncertain inferences in narrative understanding.

One very general inference rule, resembling those used in expert systems, is called `Rule Instantiation (RI)` (see section 5 for details, where the dual of `RI`, called `Goal Chaining`, is discussed as well). In some cases, this can be thought of as a general form of *modus ponens* with universal instantiation and use of multiple minor premises (instantiating the antecedent of a universally quantified conditional). However, it also allows instantiation of generic conditionals such as those above. As a simple example, `RI` allows the inference

$$\frac{(\exists x[x \text{ wolf}]) \rightarrow_{.8} [x \text{ grey}], \quad [\text{W wolf}]^{.9}}{[\text{W grey}]^{.72}}$$

The superscripted numbers are interpreted as lower bounds on *subjective* probabilities (in contrast with the objective, statistical interpretation of probabilities modifying the connective in generic conditionals).

Different generalizations may assign different subjective probabilities to the same formula. In such a case, the generalization with the logically more specific antecedent has precedence (if, in fact, one antecedent *is* more specific, i.e., entails the other). Clearly, this leads to a type of non-monotonicity.

3 Glimpses of Semantics

Episodic logic is inspired in part by Montague's intensional logic, but avoids higher-order types (cf., Chierchia & Turner 1988) and dispenses with intension operators through the use of an "inherently intensional" semantics (Schubert & Pelletier 1988).

[3] "To" is an attribute forming operator that maps diadic predicate intensions to attributes. Two other operators mapping predicates to individuals are "K" and "K1," the "kind-forming" operators (Schubert & Pelletier 1987,1988). Space limitations prevent further discussion of these operators here.

Model structures for Episodic Logic are based on a very liberal ontology of "possible individuals" $\mathcal{D}$. These include possible situations $\mathcal{S}$, propositions (i.e., possible facts) $\mathcal{P}$ and properties $\mathcal{Q}$, and kinds and collections of all of these. Our formal semantics treats episodes and more generally, situations, as a category of individuals and hence as primitive rather than as structured (as in Barwise & Perry 1983). The possible situations include possible times $\mathcal{I}$ (viewed as situations with maximal propositional content), and these in turn include possible worlds $\mathcal{W}$ (viewed as unbounded times). Moments of time and possible worlds (the usual indices of possibility) are temporally minimal and maximal time intervals respectively. Episodes are temporally bounded situations. They can also be thought of as *parts of (possible) time intervals*, in which some of the propositions true over those intervals are true over those situations, some of the propositions false over those intervals are false over those situations, and the rest are undefined.

Two disjoint relations over $\mathcal{D} \times \mathcal{S}$, namely, *Actual* and *Nonactual*, determine what entities are actual and nonactual relative to a situation. (Together, these are the *participants* in the situation.) Times are *Actual* relative to exactly one world, and *Nonactual* relative to all others.

Various algebraic structures are assumed, including a boolean lattice structure for propositions and properties and two partial orderings on situations. One of these partial orderings, $\sqsubseteq$, relates subepisodes to episodes. Subepisodes may be concurrent with, or during, the episodes of which they are a part. (The relation "s during s'" is understood in terms of the nesting of *clock-time*(s) within *clock-time*(s'). Here *clock-time* is assumed to be defined for all $s \in \mathcal{S}$, yielding real intervals as values. These clock (or calendar) times must be distinguished from times which are elements of $\mathcal{I}$, i.e., propositionally maximal situations supporting "everything that happened" over a particular clock time.) Furthermore, $\sqsubseteq$ forms a join semilattice with join operator $\sqcup$; this is understood as joining two episodes which are *Actual* relative to some common time (and hence world) into a larger episode. The second partial ordering on situations, $\preceq$, is understood as relating the propositional content of concurrent situations, and is defined by $s \preceq s'$ iff $s \sqsubseteq s'$ and *clock-time*(s) = *clock-time*(s').

Truth evaluation is done with a *context* $[\![\ ]\!]$ and a *context transformation function* (Schubert & Pelletier 1988). A context simultaneously serves the purposes of an interpretation, a variable assignment function, and a valuation function. Evaluation of a formula Φ in situation s transforms the context to a new context, written as $\Phi^s[\![\ ]\!]$, which may have new values for certain variables existentially quantified in Φ.

Some sample clauses in the recursive specification of the conditions satisfied by a "coherent" context $[\![\ ]\!]$ are the following. (We use $2 = \{0, 1\}$ as truth values. A $\rightarrow$B denotes the set of *partial* functions from A to B.)

1. If α is an individual constant or variable, $[\![\alpha]\!] \in \mathcal{D}$ or $[\![\alpha]\!]$ is undefined.

2. If π is an n-place predicate,
   $[\![\pi]\!] \in \mathcal{D}^n \rightarrow \mathcal{S} \rightarrow 2$
   (short for $\mathcal{D} \rightarrow (\mathcal{D} \rightarrow \cdots (\mathcal{D} \rightarrow (\mathcal{S} \rightarrow 2))\cdots)$),
   with $[\![\pi]\!]^{d_1,\ldots,d_n,s} = [\![\pi]\!]^{d_1,\ldots,d_n,s'}$ whenever both sides are defined and $s, s' \preceq i$ for some time $i \in \mathcal{I}$. (The notation $[\![\pi]\!]^{d_1,\ldots,d_n,s}$ denotes successive function application $[\![\pi]\!](d_1)\ldots(d_n)(s)$.)

3. If π is a predicate and α a term,
   $[\![(\pi\ \alpha)]\!] = [\![\pi]\!]^{[\![\alpha]\!]}$.

In the remaining clauses Φ and Ψ are formulas, α is a variable, and η a term.

4. $[\![\neg\Phi]\!]^s = 1$ only if for all times i during s such that *Actual*(s,i), $[\![\Phi]\!]^i = 0$
   (this becomes "iff" when s is a time)
   $= 0$ only if for some time i during s such that *Actual*(s,i), $[\![\Phi]\!]^i = 1$
   $= 0$ if for some subepisode $s' \sqsubseteq s$, $[\![\Phi]\!]^{s'} = 1$
   These conditions underdetermine $[\![\neg\Phi]\!]$ – but semantic values need not be fully determined in our theory.

5. $[\![(\exists\alpha : \Phi\Psi)]\!]^s = 1$ if for some $d \in \mathcal{D}$ such that $[\![\Phi]\!]^s_{\alpha:d} = 1$, $[\![\Phi \wedge \Psi]\!]^s_{\alpha:d} = 1$
   (This becomes "iff" when s is a time).
   $[\![\ ]\!]_{\alpha:d}$ is a context obtained from $[\![\ ]\!]$ by changing the interpretation of α to $[\![\alpha]\!] = d$, while leaving the interpretations of all other atomic symbols unaltered. We omit the remaining cases.

6. $[\![\Phi \wedge \Psi]\!]^s = 1$ iff $[\![\Phi]\!]^s = 1$ and $\Phi^s[\![\Psi]\!]^s = 1$
   Note the possible context effect of interpreting Φ on the subsequent interpretation of Ψ. Again, we omit the remaining cases.

7. $[\![\Phi * \eta]\!]^s = 1$ only if $[\![\Phi]\!]^{[\![\eta]\!]} = 1$ and *Actual*$([\![\eta]\!],s)$
   $= 0$ only if $[\![\Phi]\!]^{[\![\eta]\!]} = 0$ or *Nonactual*$([\![\eta]\!],s)$
   (These conditions become "iff" when s is a time.)

The truth conditions for "**" are not semantically defined, but are constrained by various axioms relating "**" to "*", to causal operators, and to predicate type hierarchies. Two of the most important axioms are:

- $\Box\ (\forall e[[\Phi\ **\ e] \rightarrow [\Phi\ *\ e]])$
- $\Box\ (\forall e[[\Phi\ *\ e] \rightarrow (\exists e' : [e' \text{ same-time } e][\Phi\ **\ e'])])$.

Space does not permit the discussion of the semantics of equality, λ-abstraction, predicate modifiers, predicate and sentence nominalization, quantifiers other than $\exists$, disjunction, and conditionals. Also, context change cannot be discussed in detail. However, a few comments on the semantics of generic conditionals are in order. A conditional of form

$$(\exists x_1(\exists x_2(\cdots(\exists x_k \Phi)\cdots))) \rightarrow_p \Psi$$

is evaluated essentially as in Schubert & Pelletier (1988). Roughly speaking, Ψ is evaluated in all contexts resulting from "successful" evaluation of the antecedent (where each such successful evaluation supplies different "bindings" for the existential variables). The conditional is true if at least a proportion p of these contexts render Ψ true. Thus, the semantics of such conditionals is similar to that of conditional probability statements involving random variables (see Bacchus 1988a,b).

Entailment is defined as: $\Sigma \models \Psi$ iff for all coherent contexts $[\![\ ]\!]$ and all worlds $w \in \mathcal{W}$ such that $[\![\Phi]\!]^w = 1$ for all $\Phi \in \Sigma$, $[\![\Psi]\!]^w = 1$. Some entailments following from the semantic definitions are (with "during" and "same-time" interpreted as might be expected from the above discussion):

$$[(\neg\Phi)\ *\ \eta] \models \neg(\exists e{:}[e \text{ during } \eta]\ [\Phi\ *\ e])$$

$$[(\exists x : \Phi\Psi)\ *\ \eta] \models (\exists x(\exists e{:}[e \text{ same-time } \eta]\ [[\Phi \wedge \Psi]\ *\ e]))$$

$$[[\Phi \wedge \Psi]\ *\ \eta] \models [\Phi\ *\ \eta] \wedge [\Psi\ *\ \eta]$$

We should finally remark that many uncertainties and gaps remain in our semantic theory — as in any situation theory we are aware of. For instance, we have not fully formalized the probabilistic constructs and inferences. However, what is important about our attempt is that it subsumes classical logic, provides tentative extensions in several major directions, and is sufficiently carefully formalized to make future systematic analysis and revision possible. In that sense, it is at least a step in the right direction.

4 From English to Episodic Logic

It is one thing to posit a logical form for particular sentences, but quite another to generate them systematically through a grammatical/semantic formalism. An important advantage of our representation is that it is easily computed from syntactic analyses of input sentences. For instance, the following are some of the lexical and phrase structure rules and corresponding semantic rules involved in the translation of the sentence "John realized that Mary was tired":

A. NP → Mary; Mary
B. A → tired; tired
C. AP[pred] → A; $\lambda e \lambda x[[x\ \text{A'}]\ **\ e]$
D. V[be, past] → was;
E. VP → V[be] AP[pred]; AP'
F. VP[¬t,¬e] → VP[t, ¬h];
$\lambda s \lambda x(\exists_c r : [r \text{ tense' } s]\ ((\text{VP'}\ r)\ x))$
G. S → NP VP; $\lambda s((\text{VP'}\ s)\ \text{NP'})$
H. NP → that S[tense]; (That (S' *speech-time*))
I. V[past] → realized; realize
J. VP → V[trans] NP; $\lambda e \lambda x[[x\ \text{V' NP'}]\ **\ e]$
K. S → S PUNC; (S' *speech-time*)

We now illustrate how the above sentence could be translated into our logical form by tracing the process in bottom-up order, although this would not necessarily be followed in an actual translation situation. The translation of adjective "tired" is "tired" as indicated in rule B. According to rule C, the AP[pred] "tired" is translated next as $\lambda e \lambda x[[x \text{ tired}]\ **\ e]$. Note the introduction of an episodic variable by this rule ("** e" is introduced in XP[pred] formation or in VP formation). The translation of V "was" is vacuous according to rule D. The combination of AP with V[be] is governed by rule E, according to which the translation of the VP is the translation of the AP itself, i.e., $\lambda e \lambda x[[x \text{ tired}]\ **\ e]$. Next, via rule F tense is incorporated, using lexical translation

past' = before,

with the result

$$\text{VP}[\neg\text{t},\neg\text{e}]' = \lambda s \lambda x(\exists_c r1 : [r1 \text{ before } s]\ [[x \text{ tired}]\ **\ r1]).$$

Note that rule F binds the episodic λ-variable by an $\exists_c$-quantified episodic variable. We use $\exists_c$ ("a certain") to indicate that it behaves in part like a definite and in part like an indefinite determiner. "t", "e", and "h" are features that in effect store semantic information for "tense," "episode," and "have" (perfective), respectively. The [¬t,¬e] features basically

signal that further modification of the episodic variable (e.g., by adverbials) has been closed off by existential binding. Also, notice that while existentially binding an episodic λ-variable, rule F introduces another λ-variable, which will be later bound with reference time. Then, via sentence rule G, the VP is combined with the subject NP "Mary" as follows:

$$\text{S'} = \lambda s(\exists_c r1 : [r1 \text{ before } s]\ [[\text{Mary tired}] ** r1]).$$

Now the sentence "Mary was tired" needs a reference time. Since the sentence is an embedded one, we apply rule H rather than rule K. Assuming the complementizer "that" is an operator which nominalizes its complement sentence, rule H first supplies a reference time and then combines "that" with the embedded sentence "Mary was tired," producing the following translation:

$$\text{NP'} = (\text{That } (\exists_c r1 : [r1 \text{ before } \textit{now}]\ [[\text{Mary tired}] ** r1])).$$

The verb "realized" is translated as "realize" via rule I. (Modal predications are formed with the "That" operator, and treated as ordinary predications with the nominalized proposition as object.) Next, notice how a second episodic variable is introduced by rule J, which forms a VP from a transitive verb and its NP complement. Application of rule J to V' ("realize") and NP' (the nominalized sentence above) gives the following translation:

$$\text{VP'} = \lambda e \lambda x[[x \text{ realize } (\text{That } (\exists_c r1 : [r1 \text{ before } \textit{now}]\ [[\text{Mary tired}] ** r1]))] ** e].$$

The rest of the analysis goes much the same way: tense is incorporated into VP of the main clause via rule F, subject "John" is combined with VP via rule G, and so on. Each x-variable is ultimately bound by the appropriate sentence subject, and each episodic variable by an existentially quantified episode (this happens either at the VP level or at the S level). The final translation is

$$(\exists_c r2{:}[r2 \text{ before } \textit{now}]\ [[\text{John realize } (\text{That } (\exists_c r1{:}[r1 \text{ before } \textit{now}]\ [[\text{Mary tired}] ** r1]))] ** r2]).$$

Our grammar fragment also handles many combinations of tense, aspect and temporal adverbials. The following are some of the phrase structure rules for handling perfectives, tense (in addition to rule F introduced earlier), durative adverbials and negation.

Tense and Perfective:

1. $\text{VP}[\neg e, \neg h] \rightarrow \text{VP}[e, h, \text{stat}];$
   $\lambda s \lambda x(\exists_c r{:}[r \text{ extend-to } s]\ ((\text{VP' } r)\ x))$
2. $\text{VP}[\neg e, \neg h] \rightarrow \text{VP}[e, h, \neg\text{stat}];$
   $\lambda s \lambda x(\exists_c r{:}[r \text{ before } s]\ ((\text{VP' } r)\ x))$
3. $\text{S}[\neg e, \neg h] \rightarrow \text{S}[e, h, \text{stat}];$
   $\lambda s(\exists_c r{:}[r \text{ extend-to } s]\ (\text{S' } r))$
4. $\text{S}[\neg e, \neg h] \rightarrow \text{S}[e, h, \neg\text{stat}];$
   $\lambda s(\exists_c r{:}[r \text{ before } s]\ (\text{S' } r))$
5. $\text{S}[\neg e, \neg t] \rightarrow \text{S}[t, \neg h];$
   $\lambda s(\exists_c r{:}[r \text{ tense' } s]\ (\text{S' } r))$

Notice that in our analysis simple tense has two indices, namely, event time and sentence reference time, while perfective sentences effectively have three. Reference time is the same as the speech time by default. Also note that we incorporate aspectual classes, distinguished by "stat" and "$\neg$stat" features, into the translation process. This was motivated by the observation that the translation of perfectives depends on whether a given construct is stative or not. For example, in "Mary has been living in this town for twelve years," the episode of Mary's living in a certain town extends to the speech time, whereas in a non-stative perfective like "Mary has left," the episode does not necessarily have to extend to the speech time; it must merely have occurred sometime before the speech time. Our aspectual classes are quite rudimentary compared, for instance, to Moens and Steedman's (1988) who have a very detailed (but informal) analysis of the interaction between temporal reference and aspectual classes. We follow, e.g., Richards & Heny (1982) in giving tense wide scope over VP's and adverbials or even sentences. This is in contrast with Hinrichs (1988) who gives tense scope only over the predicate that corresponds to the main verb. Also note one simplification in our translation so far: they treat nominals as expressing atemporal (non-episodic) properties.

Durative Adverbials:

6. $\text{ADVL}[\text{dura}^4] \rightarrow \text{P}[\text{for}]\ \text{NP};$
   $\lambda e[(\text{duration-of } e) = \text{NP'}]$

7. $\text{VP}[\neg\text{stat}] \rightarrow \text{VP}[\text{stat}]\ \text{ADVL}[\text{dura}];$
   $\lambda e \lambda x[(\text{ADVL' } e) \wedge$
   $[(\forall e'{:}[e' \text{ during } e]$
   $(\exists e''{:}[e'' \text{ same-time } e']\ ((\text{VP' } e'')\ x))) ** e]]$

[4] "dura" is a head feature meaning "durative."

8. VP[¬stat] → VP[¬stat] ADVL[dura];
$\lambda e \lambda x[(\text{ADVL}'\ e) \wedge$
$[(\exists s{:}[[s\ (\text{seq episode})] \wedge [s\ \text{regular}] \wedge$
$[s\ \text{span}\ e]]$
$(\forall e'{:}[e'\ \text{element-of}\ s]\ ((\text{VP}'\ e')\ x))) **e]]$[5]

9. VP[¬stat] → VP[¬stat] ADVL[dura];
$\lambda e \lambda x[(\text{ADVL}'\ e) \wedge ((\text{VP}'\ e)\ x)]$

Negation:

10. VP → V[aux] ADV[not] VP;
$\lambda e \lambda x[(\neg(\exists e'{:}[e'\ \text{during}\ e]\ ((\text{VP}'\ e')\ x))) **e]$

Again aspectual classes play a key role in our translation of sentences with durative adverbials. This is because when a non-stative S or VP combines with a durative adverbial, we often need to convert the S or VP into a repetitive activity so that the resulting phrase reads "Regularly during the interval of ADVL, S happens." For example, "John dated Mary" is a performance, i.e., non-stative, but when it combines with the durative adverbial "for two years," it would be interpreted as "John dated Mary regularly for two years" via rule 8. (On the other hand, "John dated Mary for two hours" would not have such an interpretation – *cf*, rule 9.) Similar rules can be used for durative adverbials at the sentence level.

Rules for manner and for locative adverbials as well are not hard to formulate. For example,

Manner and Locative Adverbials:

11. VP → VP ADVL[manr]; $\lambda e(\text{ADVL}'\ (\text{VP}'\ e))$
12. VP →VP ADVL[loc];
$\lambda e \lambda x[(\text{ADVL}'\ e) \wedge ((\text{VP}'\ e)\ x)]$

Meaning postulates can be supplied for manner adverbials which pass the manner adverbials introduced by rule 11 "through" the **, and attach it directly to the verbal predicate.

[5] "Seq" is a function that operates on a predicate like "episode" (or "event") and produces a new predicate that is a sequence of such things. "Regular" is a predicate that takes a sequence as an argument and is interpreted such that a regular sequence of episodes consists of episodes occurring at time intervals that conform with some norm for this type of repetitive episode. Predicate "span" indicates time relationships between an episode and a sequence of episodes roughly as follows: sequence s spans episode e, if
$[(\text{clock-time (begin } e)) + d] \geq (\text{clock-time (begin } s))$ and
$[(\text{clock-time (end } s)) + d] \geq (\text{clock-time (end } e))$,
where d is the size of the "natural period" of sequence s.

While these rules handle fairly complex cases of tense/aspect/adverbial interactions, we regard our approach to tense and aspect as an interim measure, ultimately to be replaced by a context mechanism that propagates a set of reference times "forward" and "downward" from the preceding text.

5 Making Inferences with Episodic Logic

Our inference rules fall into three broad categories, namely, (i) basic inference rules including rule instantiation and its dual goal reduction, (ii) narrative inference rules such as causal connection, temporal succession and state persistence, and (iii) simulative inference rules. All of the rules allow for subjective probability bounds on premises and for use of generic conditionals.

We have already briefly described **Rule Instantiation (RI)** which is heavily used in input-driven inference; its dual **Goal Chaining (GC)** similarly dominates goal-driven inference. We have indicated that **RI** has *modus ponens* as special case, but also allows instantiation of generic conditionals. In fact, it allows arbitrarily many "minor premises" to be matched against arbitrarily deeply embedded subformulas of a rule. (Apart from its avoidance of skolemization, it resembles Andrews' *general matings* (1981) and Bibel's *connections* (1979).) Schematically, the rule is:

RI (Rule Instantiation)

– Non-probabilistic version –

For $R(\Phi_1, \ldots, \Phi_m, \Phi'_1, \ldots, \Phi'_n)$, $\Psi_1, \ldots, \Psi_m$, $\neg\Psi'_1, \ldots, \neg\Psi'_n$, formulas with bound variables standardized apart, and with all Φ_i's occurring negatively in $R(\Phi_1, \ldots, \Phi_m, \Phi'_1, \ldots, \Phi'_n)$, and all Φ'_i's occurring positively in it:

$$\frac{R(\Phi_1, \ldots, \Phi_m, \Phi'_1, \ldots, \Phi'_n) \quad \Psi_1, \ldots, \Psi_m, \neg\Psi'_1, \ldots, \neg\Psi'_n}{R_\sigma(\top, \ldots, \top, \bot, \ldots, \bot)}$$

where substitution σ unifies the Φ_i with corresponding Ψ_i and Φ'_i with corresponding Ψ'_i. $R_\sigma(\top, \ldots, \top, \bot, \ldots, \bot)$ is then simplified to eliminate the truth values.

The substitution σ applies to certain "matchable" variables which are $\forall$-quantified by a positively occurring quantifier, or $\exists$-quantified by a negatively occurring quantifier, in $R(\Phi_1, \ldots, \Phi_m, \Phi'_1, \ldots, \Phi'_n)$ or one

of the Ψ_i. A subformula occurs positively if it lies within an even number of negations, where "$\neg$", conditional antecedents, and $\forall$-quantifier restrictions count as negation, and similarly, for "negatively occurring." Computing $R_\sigma(\top, \ldots, \top, \bot, \ldots, \bot)$ involves elimination of quantifiers of variables replaced by σ, e.g., if b is substituted for x, then $(\forall x{:}\Phi\Psi)_{b/x}$ becomes $[\Phi_{b/x} \rightarrow \Psi_{b/x}]$. Details can be found in (Schubert & Hwang 1989). A probabilistic version of **RI** results when the Ψ_i or $\neg\Psi_i'$ are allowed to have non-unit lower subjective probabilities and/or R is a generic conditional.

In practice, the rule is implemented roughly as follows. A newly inferred conclusion, corresponding to one of the Ψ_i or $\neg\Psi_i'$, is used to index to the rule R. An initial determination is then made whether the instantiation is likely to succeed, and yield a useful result. If the decision is to instantiate, then the attempt to do so is performed by a recursive algorithm applied to R, which actively seeks to find appropriate Ψ_i and $\neg\Psi_i'$ instances in the knowledge base to unify with negatively and positively occurring subformulas of R. Actually, the Ψ_i and $\neg\Psi_i'$ need not even occur explicitly in the knowledge base. They may be inferred by "specialists" for type taxonomies, temporal relations, or other special classes of relations, or by limited amount of Prolog-like backchaining.

A rule instantiation typically instantiates the complete antecedent of a rule and infers the particularized consequent. However, it may only match a part of the antecedent, or match a part of the consequent. As an example of the latter, a rule which asserts that "A person who has recently eaten a meal is unlikely to be hungry" could be instantiated with the fact that a certain individual is hungry, leading to the conclusion that he has *not* eaten for some time.

Before turning to goal-directed inference rules, we should mention λ-conversion, reverse λ-conversion,[6] and substitution of equals for equals as further deductive rules available in our logic.

For goal-directed inference (e.g., in response to questions) two general methods are available.[7] The first, **Goal Chaining (GC)**, is the dual of **RI**. For comprehensibility, we state only a special (but frequently encountered) case.

GC (Goal Chaining)

For $R(\Phi)$, Ψ standardized formulas where Φ is a positively occurring subformula of $R(\Phi)$:

$$\frac{R(\Phi),\ \Psi}{\neg R_{\overline{\sigma}}(\bot)}$$

where $\overline{\sigma}$ differs from σ (in **RI**) in that it treats variables of Ψ with positively occurring $\exists$-quantifiers or negatively occurring $\forall$-quantifiers as matchable.

Like **RI**, this is a very general chaining rule, allowing not only chaining from rule consequents to antecedents, but from *any* positively occurring subformula to the rest of $R(\Phi)$ (negated and suitably instantiated).

The second class of goal-directed methods consists of standard natural deduction rules such as proving a conditional by assuming the antecedent and deriving the consequent; or proving a negative formula by assuming the positive and deriving a contradiction; or proving a universal by proving an "arbitrary instance" of it. An interesting future possibility, in the case of proofs involving assumption-making, is to activate input-driven inferencing (primarily, **RI**) once an assumption has been made, so that its important consequences will be worked out, making it easier to complete the goal-directed proof.

In our implementation, we use a sophisticated agenda-driven control structure for goal chaining (largely borrowed from ECONET – see de Haan & Schubert 1986) with goals ranked according to estimated difficulty and new knowledge accessed via concept and topic hierarchies. The aim here is not so much theorem proving power *per se*, but the ability to get at the *relevant* knowledge in a large knowledge base.

A remaining problem is, of course, the principled handling of probabilities. The state of the art in probabilistic inference (e.g., Pearl 1988, Bacchus 1988a) is not such as to provide concrete technical tools for a logic as general as episodic logic. We are, however, successfully using a "noncircularity principle" which prevents the same knowledge from being used twice to "boost" the probability of a particular conclusion. This is done by keeping track of the support set in a probabilistic inference process. Apart from this, we use independence assumptions where there are no known dependencies, and manipulate lower probabilities in accord with the laws of probability.

Finally, we mention some narrative and simulative inference rules (yet to be implemented).

[6] Reverse λ-conversion is defined as follows: Let Φ be an expression containing an occurrence of term τ, where τ contains no occurrences of variables that are bound in Φ and free in τ. Then,

$$\Phi = (\lambda\alpha\Phi_{\alpha/\tau}\ \tau),$$

where α is any variable not occurring in Φ, and $\Phi_{\alpha/\tau}$ is the result of replacing the occurrence of τ in question by α.

[7] Only the first is currently implemented.

CC (Causal Connection)

For Φ and Ψ denoting successive sentences in a narrative:

$$\frac{[\Phi ** \eta] \circ [\Psi ** \eta']}{[\eta \text{ cause-of } \eta']\cdot^{.6}}$$

where Φ, Ψ are actions or events, but Ψ is not a volitional action.

For example, given a fragment "John greeted Mary; Mary was startled," we conclude that John's greeting is the cause of Mary's being startled, with minimal degree of belief .6. We should remark that we are in the process of reformulating narrative inference rules as generic conditionals to be used in the same way as narrative domain knowledge. For example, the above rule can be reformulated as a generic conditional along the following lines: When a text source asserts two event occurrences in succession, the text source implicates that the first event caused the second.

An example of a simulative inference rule is:

SIM (Simulative Reasoning)

For τ, an individual term; $\pi \in$ find, learn, ...; Φ and Ψ, formulas; η, an episodic term; $\mathcal{K}$, major implications[8] which are public knowledge:

$$\frac{\begin{array}{c}[[\tau\ \pi\ \Phi] ** \eta] \\ \{\Phi\} \cup \mathcal{K} \vdash_f \Psi\end{array}}{\begin{array}{l}(\exists e{:}[[e \text{ right-after } \eta] \wedge [\eta \text{ cause-of } e]] \\ \quad [[\tau \text{ infer (That } \Psi)] ** e])\end{array}}$$

where $\vdash_f$ means "follows automatically by input-driven inference." Note that in this "simulative reasoning" rule, the question answerer's *own* ability to infer Ψ from Φ via major implications is being attributed to τ, the agent in the antecedent.

6 Episodic Logic and Narrative Understanding

In outline, story understanding on our view involves the following interleaved steps for each new sentence: (1) parsing and logical translation; (2) disambiguation (including quantifier scoping and anaphora resolution); (3) application of all three types of inference rules to the translated input, in combination with stored knowledge (meaning postulates, generic conditionals, and other general and specific knowledge); among other things, this may generate new predictions and explanations; (4) matching of previous predictions and explanations with new ones.

We have put these ideas to the test in two ways: first, by hand-simulating the inference process for a small fragment of the story of *Little Red Riding Hood.* We also have a prototype implementation which accepts logical-form inputs and performs many of the inferences we have alluded to and is able to answer simple questions (Schubert *et al.* 1988). In this section, we will show how our logic would allow the system to account for the wolf's decision not to eat Little Red Riding Hood right away when he first met her, given a brief excerpt from the story as follows.

In the forest, Little Red Riding Hood met a wolf. The wolf would have very much liked to eat her, but he dared not do so on account of some woodcutters nearby.

Processing this fragment requires extensive reasoning including inferences based on meaning postulates, predictive inferences, explanatory inferences and simulative inferences. For example, to understand the third sentence, one should be able to explain why the wolf decided against eating Little Red Riding Hood, and how the presence of woodcutters nearby affected the wolf's decision. So, one has to know that when some agent *dares* not do something, he must think it possible that his attempt to do it would result in something unpleasant to himself; then one has to simulate his reasoning process to guess what unpleasant consequences he anticipates.

Depending on the degree of sophistication of the knowledge possessed, people may explain the wolf's decision in various ways. Correspondingly, depending on the kind of knowledge provided, our inference machinery can produce various lines of reasoning; this includes the following, relatively simple line of reasoning

- Attacking a child is extremely wicked.
- Trying to eat a living creature involves attacking it, and such an attack is conspicuous and likely to be noticed by nearby people.
- Doing something extremely wicked is likely to bring severe punishment, if noticed by anyone.
- So, if the wolf tries to eat Little Red Riding Hood, the nearby woodcutters may notice it, and he is likely to be severely punished for it.

[8] Major implications are the ones that people generally use for "input-driven" inferences.

Or, the more sophisticated version

- When a predatory animal eats a non-predatory creature of comparable size while the creature is conscious, the predator attacks it as a preparation for eating it.
- The wolf would attack Little Red Riding Hood before eating her.
- Attacking a person is a conspicuous action, and is likely to be noticed by nearby people.
- If people notice a predatory animal attacking a person, they will most probably want to rescue the person from the animal.
- To rescue a person from a predatory animal, one may kill it.
- Thus, the woodcutters may kill the wolf.

Upon reaching the conclusion that it is possible that the wolf might be killed or severely punished, our inference machinery attributes its own ability to infer that conclusion to the wolf (this is due to our rules of simulative reasoning). Then it is easily explained why the wolf decided against eating Little Red Riding Hood right then and there.

As space limitations do not allow the inclusion of a detailed analysis, we show in the following only that part of the reasoning process reaching the conclusion "The wolf may be severely punished." In working out the inferences by hand, we assumed a control structure which systematically combines each new clause with relevant meaning postulates and other general knowledge, and makes relevant narrative inferences. All of these inferences are based on the explicit, formalized rules of inference we introduced earlier. After listing meaning postulates and world knowledge, we show the logical translation of the story and the reasoning process.

Meaning Postulates

M1. To walk, to attack someone, to try to do something, to die, etc., are types of actions.

For Φ an action formula:

$\Box$ [(To $\lambda e \lambda x \Phi$) action-type]

"To" is an operator forming an action type from a two-place predicate. An action formula is of form $[[x\ \pi] **e]$, where x and e are free, and π is an action predicate; an action predicate is an expression (P $\tau_1, \ldots, \tau_{n-1}$), where P is an n-adic atomic action predicate, and $\tau_1, \ldots, \tau_{n-1}$ are terms.

M2. If there is a collection of things of some type, then there is a thing of that type which belongs to that collection (we regard collections as non-empty by definition).

For P a monadic predicate:

$\Box$ $(\forall x{:}[x\ (\text{coll P})]\ (\exists y{:}[y\ \text{in}\ x][y\ \text{P}]))$

"coll" is a function that maps a predicate applicable to things into a predicate applicable to collections of things.

\# An "action" is an (ordered) event-agent pair, "instantiating" or "realizing" the action type (To $\lambda e \lambda x \Phi$), for some Φ. We use "|" for the pairing function, e.g., (| e x), where e is an event and x is an agent. We also allow the infix notation $[e \mid x]$ instead of (| e x).

For pair p:
(fst p) returns the first of the pair;
(rst p) returns the rest of the pair.

Some meaning postulates regarding actions (Φ is an action formula as in M1):

M3. $\Box$ $(\forall a[[a\ \text{instance-of (To}\ \lambda e \lambda x \Phi)]$
$\leftrightarrow((\lambda e \lambda x \Phi\ (\text{fst}\ a))\ (\text{rst}\ a))]\ )$

For example,
[[E1 | John] instance-of (To $\lambda e \lambda x[[x\ \text{eat}] **e]$)]
$\leftrightarrow((\lambda e \lambda x[[x\ \text{eat}] **e]\ \text{E1})\ \text{John})$
i.e., $\leftrightarrow$[[John eat] ** E1]

M4. M3 can be equivalently expressed as:

$\Box$ $(\forall e(\forall x[[[e \mid x]\ \text{instance-of (To}\ \lambda e' \lambda x' \Phi)]$
$\leftrightarrow((\lambda e' \lambda x' \Phi\ e)\ x)]\ ))$

M5. For π an action predicate:

$\Box$ $(\forall e(\forall x[[[x\ \pi] **e] \rightarrow[x\ \text{do}\ [e \mid x]]]))$

(Remark: Note that [x do [e | x]] is not itself a description of a bounded episode; rather, it is an "eternal" truth, if true at all. It can be understood equivalently as saying that [e | x] *is* an action, not just an arbitrary event-individual pair, so that x *is* the agent of that action.)

World Knowledge

K1. For a creature to attack a child is extremely wicked.

$(\exists x{:}[x\ \text{creature}](\exists y{:}[y\ \text{child}](\exists e[[x\ \text{attack}\ y] **e])))$
$\rightarrow_{.9}$ [[e | x] (extremely wicked)]

K2. Trying to eat any living creature involves attacking it.

(∀*y*:[[*y* alive] ∧ [*y* creature]]
  [(To λ*e*λ*x*[[*x* try
    (To λ*e'*λ*x'*[[*x'* eat *y*] ** *e'*])] ** *e*])
  involve (To λ*e*λ*x*[[*x* attack *y*] ** *e*])])

K3. If one type of action involves another, then any creature doing an instance of the first will do an instance of the second during it.

(∀*x*:[*x* creature]
  (∀*a*1:[*a*1 action-type]
    (∀*a*2:[[*a*2 action-type] ∧ [*a*1 involve *a*2]]
      (∀*e*1:[[*e*1 | *x*] instance-of *a*1]
        (∃*e*2:[*e*2 during *e*1]
          [[*e*2 | *x*] instance-of *a*2])))))

K4. For a sizable creature to attack a sizable thing is conspicuous (relative to a human observer).

(∃*x*:[*x* person]
  (∃*y*:[[*y* creature] ∧ ¬[*y* tiny-rel-to *x*]]
    (∃*z*:[[*z* creature] ∧ ¬[*z* tiny-rel-to *y*]]
      (∃*e*[[*y* attack *z*] ** *e*]))))
  →.9 [[*e* | *y*] conspicuous-to *x*]
By contrast, for an ant to attack something would not be conspicuous to a human.

K5. If a creature performs a conspicuous action within plain sight of a person, that person is likely to notice that action.

(∃*x*:[*x* creature]
  (∃*y*:[*y* person]
    (∃*e*1:[[*x* within-plain-sight-of *y*] ** *e*1]
      (∃*e*2:[*e*2 during *e*1]
        [[*x* do [*e*2 | *x*]] ∧
          [[*e*2 | *x*] conspicuous-to *y*]])))))
  →.6 (∃*e*3:[*e*3 during *e*2] [[*y* notice [*e*2 | *x*]] ** *e*3])

K6. Doing something extremely wicked may bring severe punishment from some group of people, if noticed by anyone.

(∃*x*:[*x* creature]
  (∃*y*:[*y* person]
    (∃*e*1:[[*x* do [*e*1 | *x*]] ∧
        [[*e*1 | *x*] (extremely wicked)]]
      (∃*e*2 [[*y* notice [*e*1 | *x*]] ** *e*2])))))
  →.3 (∃*p*:[*p* (coll person)]
    (∃*e*3:[*e*2 cause-of *e*3]
      [[*p* (severely (punish *x*))] ** *e*3]))

K7. A human is not tiny relative to a wolf, and vice versa.

(∀*x*:[*x* human] (∀*y*:[*y* wolf]
  [¬[*x* tiny-rel-to *y*] ∧ ¬[*y* tiny-rel-to *x*]]))

K8. If a creature is near a person and not tiny relative to the person, it is probably within plain sight of the person. (This could be improved by assuming that we are dealing with a *daytime* episode in an open setting.)

(∃*x*:[*x* person]
  (∃*y*:[[*y* creature] ∧ ¬[*y* tiny-rel-to *x*]]
    (∃*e*1 [[*y* near *x*] ** *e*1])))
  →.6 (∃*e*2:[*e*2 same-time *e*1]
    [[*y* within-plain-sight-of *x*] ** *e*2])

K9. Woodcutters are humans.

(∀*x*:[*x* woodcutter] [*x* human])

Story

Now, let's work out the possible consequences if the wolf tries to eat Little Red Riding Hood. (We then attribute this reasoning to the wolf). The relevant assumptions and story facts are as follows (where we use the convention of having variables in lower case, and constants in upper case):

\# The wolf tries to eat Little Red Riding Hood.

(∃e1:[*now* during e1]
  (∃x1:[x1 wolf] [[x1 try
    (To λ*e*λ*x*[[x eat LRRH] ** e])] ** e1]))

By Skolemizing {E1/e1, W/x1}:

S1. [*now* during E1]
S2. [W wolf]
S3. [[W try (To λ*e*λ*x*[[x eat LRRH] ** e])] ** E1]

Note that S3 can be rewritten as follows by two reverse λ-conversions:
((λe'λx'[[x' try
  (To λ*e*λ*x*[[x eat LRRH] ** e])] ** e'] E1) W)

\# Little Red Riding Hood is a girl and alive.

S4. [LRRH girl]
S5. [LRRH alive]

There are woodcutters nearby.

($\exists y$:[y (coll woodcutter)]
($\forall x$:[x in y]
($\exists e2$:[E1 during $e2$] [[W near x] ** $e2$])))

By Skolemizing {C1/y}:

S6. [C1 (coll woodcutter)]
S7. ($\forall x$:[x in $C1$]
($\exists e2$:[E1 during $e2$] [[W near x] ** $e2$]))

Assume the following type-hierarchical knowledge is available (at least indirectly, via a type "specialist"):

S8. [W creature]
S9. [LRRH child]
S10. [LRRH human]
S11. [LRRH creature]

Reasoning Process

- Note that simple time inferences such as
[E1 during E2] ∧ [E2 during E3] ∧
[E3 same-time E4] ⊢ [E1 during E4]
will be taken for granted during the inference process.

- In the following,
RI [A; B] {Subst C/v; Imm-Skol C'/v'}
indicates that the subsequent inference(s) has been made via **Rule Instantiation** of rule B by premise(s) A, with variable substitution C/v, and an existential variable v' in the inferred formula has been immediately skolemized as C'.

RI [S5, S11; K2] {Subst LRRH/y}:
1. [(To λeλx[[x try
(To λe'λx'[[x' eat LRRH] ** e'])] ** e])
involve (To λeλx[[x attack LRRH] ** e])]
"Trying to eat LRRH involves attacking her."

RI [; M1] {Subst [[x try (To λe'λx'[[x' eat
LRRH] ** e'])] ** e]/Φ}:
2. [(To λeλx [[x try (To λe'λx'[[x' eat
LRRH] ** e'])] ** e]) action-type]
"Trying to eat LRRH is an action type."
RI [; M1] {Subst [[x attack LRRH] ** e]/Φ}:
3. [(To λeλx[[x attack LRRH] ** e]) action-type]
"Attacking LRRH is an action type."

RI [S3; M4] {Subst E1/e, W/x, [[x try (To λe'λx'[[x'
eat LRRH] ** e'])] ** e]/Φ}:
4. [[E1 | W] instance-of (To λeλx [[x try
(To λe'λx'[[x' eat LRRH] ** e'])] ** e])]
"The wolf's trying to eat LRRH is an instance of someone's trying to eat LRRH."

RI [S8, 2, 3, 1, 4; K3] {Subst W/x, E1/e1,
(To λeλx [[x try (To λe'λx'[[x' eat
LRRH] ** e'])] ** e])/a1,
(To λeλx[[x attack LRRH] ** e])/a2;
Imm-Skol E2/e2}:
5. [E2 during E1]
6. [[E2| W] instance-of
(To λeλx[[x attack LRRH] ** e])]

RI [6; M4] {Subst E2/e, W/x, [[x attack LRRH]/Φ}:
7. [[W attack LRRH] ** E2]
"The wolf will attack LRRH."

RI [7; M5] {Subst E2/e, W/x}:
8. [W do [E2 | W]]
"The wolf's attack is an action."

RI [S8, S9, 7; K1] {Subst W/x, LRRH/y, E2/e}:
9. $[[E2 \mid W] \text{ (extremely wicked)}]_{.9}$
Up to here:
**The wolf will attack LRRH,
and that's extremely wicked.**

RI [S10, S2; K7] {Subst LRRH/x, W/y}:
10. ¬[LRRHtiny-rel-to W]
"LRRH is not tiny relative to the wolf."

RI [S6; M2] {Subst C1/x, woodcutter/P;
Imm-Skol C2/y}:
11. [C2 in C1]
12. [C2 woodcutter]
"There is a woodcutter."

RI [12; K9] {Subst C2/x}:
13. [C2 human]
"The woodcutter is a human."

With type-hierarchical knowledge, we get from 13:
14. [C2 person]

RI [13, S2; K7] {Subst C2/x, W/y}:
15. ¬[W tiny-rel-to C2]
"The wolf is not tiny relative to the woodcutter."

RI [14, S8, 15, S11, 10, 7; K4]
{Subst C2/x, W/y, LRRH/z, E2/e}:
16. $[[E2 \mid W] \text{ conspicuous-to } C2]_{.9}$

Up to here:
The wolf's attack will be conspicuous to the woodcutter.

RI [11; S7] {Subst C2/x; Imm-Skol E3/e}:
17. [E1 during E3]
18. [[W near C2] ** E3]
"The wolf is near the woodcutter (when he tries to eat LRRH)."

RI [14, S8, 15, 18; K8]
{Subst C2/x, W/y, E3/e1; Imm-Skol E4/e2}:
19. [E4 same-time E3]
20. [[W within-plain-sight-of C2] ** E4]$_{.6}$
"The wolf is likely to be within plain sight of the woodcutter."

RI [S8, 14, 20, (5, 17, 19), 8, 16; K5] {Subst W/x, C2/y, E4/e1, E2/e2; Imm-Skol E5/e3}:
21. [E5 during E4]
22. [[C2 notice [E2 | W]] ** E5]$_{.324}$
Up to here:
The woodcutter may notice the wolf's attacking LRRH.

RI [S8, 14, 8, 9, 22; K6] {Subst W/x, C2/y, E2/e1, E5/e2; Imm-Skol C3/p, E6/e3}:
23. [C3 (coll person)]
24. [E5 cause-of E6]
25. [[C3 (severely (punish W))] ** E6]$_{.087}$
The wolf may be severely punished by some group of people.

This inference chain can be extended to provide an explanation for the wolf's decision *not* to try to eat LRRH at that point in the story. First, rule K3 would be slightly augmented so as to express the fact that if one action involves another, and that other action has certain consequences, then these are also consequences of the first action. Rule K5 would be similarly augmented to make the "noticing event" e3 a causal consequence of the event e2 (or action [e2 | x]) noticed. *The "punishing event " E6 in conclusion 25 would then be inferred to be a consequence of the wolf's attempt to eat LRRH* . Given that being severely punished is very bad, and that agents generally refrain from actions that they think may have very bad consequences for them, we would have an explanation for the wolf's restraint. Note, however, that this requires application of the simulative inference rule (**SIM**), i.e., we must attribute the above inference chain to the wolf, and draw further conclusions from this attribution.

7 Concluding Remarks

Our logic is probably the most expressive yet brought to bear on the problem of narrative understanding. It makes implicit time and situation dependencies explicit through the use of episodic variables, and admits unbound "anaphoric" variables and the representation of generic conditionals.

Our use of episodic variables owes a debt to Davidson (1969), but we can "attach" an episodic variable to any formula, whereas Davidson's method can introduce episodes only for atomic formulas. Thus, for Davidson, there can be no episodes involving *quantification*, such as an episode of everyone in the room looking at Mary, and no episodes involving *negation*, such as an episode of John not eating anything for ten hours. Yet such episodes can perfectly well be cited as causal antecedents (e.g., sentences (4) and (5) in section 2; or "John did not eat anything for ten hours, and as a result, he was famished"), anaphorically referred to, quantified over, etc.

We should also mention the Situation Calculus of McCarthy and Hayes (1969), whose notion of a situation corresponds exactly to our notion of a (possible) moment of time, and the Event Calculus of Kowalski (1986), which treats events as individuals as we do. Like Davidson, however, Kowalski is unable to deal with events involving quantification and logical compounds.

We have provided evidence that our episode-based logical form can provide a clean foundation for story understanding. The main advantages of our approach are as follows:

(a) The representation of phrase structure is modular and transparent, as is the mapping from phrase structure to Episodic Logic. The mapping handles many combinations of tense, aspect and adverbials.

(b) Episodic Logic is expressively rich — it allows the content of most English sentences and most world knowledge to be represented in an intuitively comprehensible and formally analyzable manner. Restricted quantifiers, modal operators, nominalization operators, episodic variables, anaphoric variables, and generic conditionals are brought together for the first time in a logic for narrative understanding.

(c) Being probabilistic, our rules of inference allow evidence for explanations or predictions to be weighed, much as in expert systems.

(d) All types of linguistic and domain knowledge are strictly separated from parsing and inference control structure, allowing the former to be expanded and revised independently of the latter.

(e) Hand-simulation of the processing of actual story fragments, and question-answering, indicates that our logical framework is epistemologically adequate for story understanding.

This last claim, about epistemological adequacy, may come as something of a surprise. Whatever happened to scripts, plans, TAUs, TOPs, MOPs, etc.? Are these "higher-level" knowledge structures not essential to story comprehension? We do not doubt that they are. However, we see no sharp divisions between any of them. The more focused the successive stages of a script are on an ultimate goal, the more it resembles a plan. The more abstract its level of description, the more it resembles a TAU or a TOP, and so on. Furthermore, we see no particular obstacle to encoding all of them as axiomatic knowledge in episodic logic, in the manner of the examples in section 2. For example, the M-BORROW MOP (Dyer 1983: 207) can be cast as a set of generic conditionals along the following lines. If some person x wants to have some object y temporarily, which he knows to be in the possession of some person z, he may well ask z to lend him y and this may induce z to do so, fulfilling x's goal. If some person x has some object y on loan from some person z, then x is obligated to return y to z, and z will probably want him to do so; etc. We consider the taxonomy of scripts, plans, MOPs, etc., and their elaborate subcategorization, more of a potential guide to control structure – what knowledge is likely to be useful when – than a guide to representation.

Much work remains to be done on our logic – for instance, on the formal semantics of nominalization and propositional attitudes, and of probabilistic inference. As well, we need to compose many more rules of translation and compile a substantial body of knowledge for particular stories. However, our implementation to date has proved to be very gratifying (ECOLOGIC: Schubert *et al.* 1988), and we have incorporated several techniques into ECOLOGIC which were developed for an earlier system based on ordinary first-order logic (ECOSYSTEM: de Haan & Schubert 1986, Miller *et al.* 1987, Miller & Schubert 1988), facilitating efficient deduction (both general and specialized) and fast, selective access to knowledge relevant to a particular set of concepts and topics. "Natural" goal reduction remains to be implemented, but nonetheless the types of questions handled by ECONET are also handled by ECOLOGIC(e.g., "Did anyone have some cake?" or "Does grandmother live in a shoe?")

Acknowledgements

The authors are indebted to Philip Harrison for many detailed criticisms and suggestions. Stephanie Miller has been the mainstay of the program development effort for ECOLOGIC. As well, James Allen provided helpful comments, and the University of Alberta Logical Grammar Study Group provided a forum for general discussion of the topic. This research was supported in part by NSERC Operating Grant A8818 (LKS), an Izaak W. Killam Memorial Scholarship (CHH), and Boeing Co. under Purchase Contract W-278258.

References

[Andrews, 1981] P. B. Andrews, "Theorem proving via general matings," *JACM*, 28(2):193–214, 1981.

[Bacchus, 1988a] F. Bacchus, *Representing and Reasoning with Probabilistic Knowledge*, PhD thesis, U. of Alberta, Edmonton, Alberta, 1988.

[Bacchus, 1988b] F. Bacchus, "Statistically founded degrees of belief," In *Proc. of the 7th Bienn. Conf. of the Can. Soc. for Computational Stud. of Intelligence (CSCSI '88)*, pages 59–66, Edmonton, Alberta, June 6-10 1988.

[Barwise and Perry, 1983] J. Barwise and J. Perry, *Situations and Attitudes*, MIT Press (Bradford Book), Cambridge, Mass., 1983.

[Bibel, 1979] W. Bibel, "Tautology testing with a generalized matrix reduction method," *Theor. Comput. Sci.*, 8:31–44, 1979.

[Chierchia and Turner, 1988] G. Chierchia and R. Turner, "Semantics and property theory," *Linguistics and Philosophy*, 11:261–302, 1988.

[Davidson, 1969] D. Davidson, "The individuation of events," In N. Rescher et al., editor, *Essays in Honor of Carl G. Hempel*, pages 216–234. Reidel, Dordrecht, Holland, 1969.

[de Haan and Schubert, 1986] J. de Haan and L. K. Schubert, "Inference in a topically organized semantic net," In *Proc. AAAI-86*, pages 334–338. Philadelphia, PA., Aug. 11-15 1986.

[Dyer, 1983] M. G. Dyer, *In-Depth Understanding*, MIT Press, Cambridge, Mass., 1983.

[Heim, 1982] I. Heim, *The Semantics of Definite and Indefinite Noun Phrases*, PhD thesis, U. of Mass., 1982.

[Hinrichs, 1988] E. W. Hinrichs, "Tense, quantifiers, and contexts," *Computational Linguistics*, 14(2):3–14, 1988.

[Hurum, 1987] S. Hurum, "Quantifier scoping in initial logical translations of English sentences," Master's thesis, U. of Alberta, Edmonton, Alberta, 1987.

[Hurum, 1988] S. Hurum, "Handling scope ambiguities in English," In *Proc. 2nd Conf. on Applied Natural Language Processing*, pages 58–65. Austin, Texas, Feb. 9-12 1988.

[Hurum and Schubert, 1986] S. Hurum and L. K. Schubert, "Two types of quantifier scoping," In *Proc. 6th Canadian Conf. on Artificial Intelligence (AI-86)*, pages 39–43. Montreal, May 21-23 1986.

[Kamp, 1981] H. Kamp, "A theory of truth and semantic representation," In J. Groenendijk, T. Janssen, and M. Stokhof, editors, *Formal Methods in the Study of Language*. Mathematical Centretracts, U. Amsterdam, Amsterdam, 1981.

[Kolodner, 1981] J. L. Kolodner, "Organization and retrieval in a conceptual memory for events or CON54, where are you?," In *Proc. IJCAI-81*, volume 7, pages 227–233. Vancouver, August 24-28 1981.

[Kowalski, 1986] R. Kowalski, "Database updates in the Event Calculus," Technical Report DOC 86/12, Dept. of Computing, Imperial College, London, 1986.

[Lehnert *et al.*, 1983] W. G. Lehnert, M. Dyer, P. Johnson, C. Yang, and S. Harley, "BORIS – An experiment in in-depth understanding of narratives," *Artificial Intelligence*, 20:15–62, 1983.

[McCarthy and Hayes, 1969] J. McCarthy and P. J. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," In B. Meltzer et al., editor, *Machine Intelligence*, 4, pages 463–502. Edinburgh U. Press., 1969.

[Miller *et al.*, 1987] S. Miller, J. de Haan, and L. K. Schubert, *The User's Guide to ECoNet* (Prepared for Boeing Co. under Purchase Contract W-278258), Edmonton, Alberta, 1987.

[Miller and Schubert, 1988] S. Miller and L. K. Schubert, "Using specialists to accelerate general reasoning," In *Proc. AAAI-88*, pages 161–165, St. Paul, Minn., August 21-26 1988.

[Moens and Steedman, 1988] M. Moens and M. Steedman, "Temporal ontology and temporal reference," *Computational Linguistics*, 14(2):15–28, 1988.

[Pearl, 1988] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufman, San Mateo, CA, 1988.

[Richards and Heny, 1982] B. Richards and F. Heny, "Tense, aspect, and time adverbials, Part I," *Linguistics and Philosophy*, 5:59–107, 1982.

[Schubert and Hwang, 1989] L. K. Schubert and C. H. Hwang, "A Logical Approach to Narrative Understanding," Technical report, 1989, in preparation.

[Schubert *et al.*, 1988] L. K. Schubert, S. Miller, and C. H. Hwang, *The User's Guide to ECOLOGIC* (Prepared for Boeing Co. under Purchase Contract W-278258), Edmonton, Alberta, 1988.

[Schubert and Pelletier, 1982] L. K. Schubert and F. J. Pelletier, "From English to logic: context free computation of 'conventional' logical translations," *American J. of Comp. Ling.*, 8:26–44, 1982, Also in *Readings in Natural Language Processing*, B. Grosz, K. Jones and B. Webber, eds., 293-311, Morgan Kaufman, Los Altos, Calif., 1986.

[Schubert and Pelletier, 1987] L. K. Schubert and F. J. Pelletier, "Problems in the interpretation of the logical form of generics, bare plurals, and mass terms," In E. LePore, editor, *New Directions in Semantics*, pages 387–453. Academic Press, London, 1987.

[Schubert and Pelletier, 1988] L. K. Schubert and F. J. Pelletier, "Generically speaking, or, using discourse representation theory to interpret generics," In G. Chierchia, B. Partee, and R. Turner, editors, *Property Theory, Type Theory, and Semantics*. Reidel, Dordrecht, Holland, 1988, in press.

Syntactic Equality in Knowledge Representation and Reasoning

Edward P. Stabler, Jr.*
Canadian Institute for Advanced Research, and
Department of Computer Science
University of Western Ontario
London, Canada

Abstract

Reasoning with equality can be computationally difficult, but is less so when the equality relation for a language is the familiar relation of "syntactic equality:" no two distinct ground terms denote the same thing in the intended model. However, first order theories often contain existentially quantified variables which block the use of the syntactic equality methods. The existence of a decidable set of terms known to have pairwise distinct denotations can nevertheless be a computational advantage. We present an extension of resolution, "SEq resolution," that builds in the syntactic equality axioms for just those terms known to be pairwise distinct. In some cases, there is a computational advantage to respecting the syntactic equality restriction in so far as possible, leaving only Skolem functions and constants as the exceptions, and using SEq resolution.

1 Introduction

In a recent book by Wos et al. [1984], the art of automated reasoning is codified in a number of useful proverbs. One of them is the following:

Employ equality predicates.

This recommendation is surprising to logic programmers who assiduously avoid equality and other congruence (or even equivalence) relations. Consider the following remark from the logic programming literature:

> *One of the reasons why logic programming succeeded where other resolution theorem proving had failed (namely in finding a practical application) was that in logic programming equality was avoided like the plague.*[van Emden and Yukawa , 1980]

Some knowledge is very hard to represent without any equality, though. Consider expressing the simple fact that the only things that have property p are a and b. With equality, this is easily expressed with a formula like $p(X) \leftrightarrow X = a \vee X = b$. Logic programmers have noticed that a very restricted, computationally manageable equality relation suffices to express such things: the familiar "syntactic equality." This relation is so restricted that complete proof methods using it do not need to consider transitive chains of equations or substitutions of distinct terms into any predicate or function expressions. In this paper we generalize this idea to full first order resolution. It is a simple matter to build the syntactic equality theory for any first order language into an inference rule. In fact, we propose a more general idea: access to an efficiently decidable set of terms whose denotations are known to be pairwise distinct can be a real computational advantage. This paper shows how extensions of resolution and similar strategies can exploit such a set of terms for computational advantage even when the set does not include all the terms in the theory. We propose a more specific version of the logic programmers' maxim:

> *Avoid non-syntactic equality. That is, try to minimize the number of terms that are not in the set of terms whose denotations are known to be pairwise distinct.*

An analysis of some examples that support this recommendation about knowledge representation confirm what becomes clear with a little reflection on the matter. The apparent tension between our proverb and Wos's is explained by the observation that while non-syntactic equality can simplify theories and proofs, it can make the search for a proof very difficult. When the search is guided, simplicity is preferable, though it is of course still nice to have a small search space – neither humans nor machines are very good at considering too many alternatives. When search is auto-

*I am grateful to the Canadian Institute for Advanced Research and the Natural Sciences and Engineering Research Council of Canada for supporting this research. I would like to thank William Demopoulos for valuable discussions of this material. I am also grateful to Hector Levesque, Alan Mackworth and Ray Reiter for convincing me, some time ago, that I should get clear about this stuff.

matic, a small search space is essential for practical performance. Syntactic equality provides the expressive power needed to make a theory and deductions from it simple, and yet minimizes the search space.

2 A First Example

Consider the following simple theory:

$nn(0)$ *Theory HNN*
$nn(X) \rightarrow nn(s(X))$

where, here and throughout, we use such formulas to represent their universal closures, and variable names begin with uppercase letters. Under the obvious interpretation, **N**, this theory says (truly) that 0 and its successors are natural numbers. Suppose that the language contains just one other function symbol, the constant a, and in **N** neither a nor any of its successors names a natural number. Then our theory has the nice properties that $HNN \not\models nn(a)$, and in fact, $HNN \vdash nn(\tau)$ just in case τ names a natural number in **N**. Obviously, though, $HNN \not\models \neg nn(a)$. To get the latter entailment we need, in the first place, a necessary condition for natural numberhood like the following:

Theory NN
$nn(X) \leftrightarrow (X = 0 \vee \exists Y(X = s(Y) \wedge nn(Y)))$

And then we need a theory of equality that tells us that our terms are pairwise distinct, and in particular that the term a does not have the same denotation as any other ground term:

$X = X$ *Theory SEq(NN)*
$a \neq 0$
$a \neq s(X)$
$0 \neq s(X)$
$X \neq \tau$ *for all terms τ properly containing X* (†)
$s(X) = s(Y) \leftrightarrow X = Y$
$(nn(Y) \wedge X = Y) \rightarrow nn(X)$
$(Y_1 = Y_2 \wedge X_1 = Y_1 \wedge X_2 = Y_2) \rightarrow X_1 = X_2$

Clearly, we now have $NN \cup SEq(NN) \models \neg nn(a)$, and $NN \cup SEq(NN) \models \neg nn(s(a))$.

Notice that this theory contains a schema, (†), with infinitely many instances. In fact this equality theory has a form that is familiar in the logic programming literature: $NN \cup SEq(NN)$ is essentially Clark's *completion* of the definite clause theory *HNN* [Clark, 1978] [Lloyd, 1984]. Clark uses a completion like this in his characterization of the results that can be obtained from *HNN* when Horn clause resolution is extended with the non-monotonic negation-as-failure rule. We do not need to tangle with negation-as-failure, though.[1] Classical logic gives us entailments like those mentioned, and a sound and complete method like resolution allows us to demonstrate such results.

For present purposes, the interest of equality theories like $SEq(NN)$ is that they serve to force a "free" interpretation of the terms in the language. In other words, the terms of the theory satisfy a "unique names condition," distinct terms (distinct names *and* distinct ground function expressions generally) have distinct denotations. However, the well known proof techniques do not allow us to obtain the computational advantage of syntactic equality in deductions from $NN \cup SEq(NN)$ because the theory contains an existential quantifier. When converted to clausal form, a Skolem function will be introduced for which the syntactic equality restriction cannot be presumed to hold. We will show how to take advantage of the fact that all the other terms respect the unique names assumption. We will briefly describe a very simple sound and complete specialized inference rule for syntactic equality, and then modify it to handle theories like the clausal form of $NN \cup SEq(NN)$ in which a subset of the terms fail to satisfy the syntactic equality restriction.

3 Building in syntactic equality

Because equality vastly increases the search space of a theorem prover, there has been a good deal of work on building the standard equality axioms Eq into special inference rules R in such a way that $T \vdash_R S$ iff $T \cup Eq \models S$. If $T \cup Eq$ is satisfiable, we say T is *E-satisfiable*. A rule R meeting the above condition is said to be a sound and complete method for E-unsatisfiability. Paramodulation is probably the best known method [Chang and Lee, 1973] [Wos *et al.*, 1980]: equalities license the substitution (or "paramodulations") of distinct terms in any literal at any point in the deduction. We can use a similar technique to build in the equality axioms that enforce the unique names condition. We have seen an example of these axioms in the case of $SEq(NN)$, above. In general, let the *axiomatization of the unique names assumption $SEq(T)$ for the language of T* be defined to comprise the standard equality axioms Eq for the language, pairwise inequalities for all the constants and function symbols, all instances of †, and for each function symbol an axiom saying that it is a 1-1 function. Then we say that T is *SEq-satisfiable* iff $T \cup SEq(T)$ is satisfiable. Obviously, then, $NN \cup nn(a)$ is SEq-unsatisfiable.[2]

Building in the (often infinitely many) axioms of SEq, we can formulate a simple, clausal refutation method for SEq-unsatisfiability. The sufficient condition for identity in our restricted models is easily expressed. It can be represented simply by the clause:

[1] For a consideration of some of the troubles with negation-as-failure in logic programming, see, for example, [Flannagan, 1986] [Sheperdson, 1984].

[2] We modify the definition of SEq-satisfiability below to handle the case where not all terms satisfy the unique names condition.

$\{X = X\}$. A resolution step in which this clause plays a role can be easily replaced by a rule that has the same effect – our rule (i) below. The necessary condition on syntactic equality is the tricky part. We want to say that two things are nonidentical just in case they are named by distinct terms. The following simple extensions to resolution suffice:

For any clause C,

(i) If $t \neq t' \in C$ and σ is the mgu of t and t', deduce $(C - \{t \neq t'\})\sigma$;

(ii) If $t = t' \in C$ and for substitution σ, $t\sigma$ and $t'\sigma$ are not unifiable, deduce $(C - \{t = t'\})\sigma$.

Theorem 3.1. There is a derivation of $\square$ from Γ using resolution together with (i) and (ii) iff Γ is not SEq-satisfiable.

Proof: We must show $\Gamma \vdash \square$ iff Γ is SEq-unsatisfiable. The result for derivations that do not unify literals containing the equality predicate was established by Robinson [Robinson, 1965], so we consider only our special treatment of equality.

When a clause C contains a literal $\neg t = t'$ we allow the deduction of $C - \{\neg t = t'\}\sigma$ just in case σ is the most general unifier of t and t'. This is sound since $X = X$ is true, and it is complete because two terms denote the same thing in a model that satisfies our semantic restriction only when the terms are identical.

When a clause C contains a literal $t = t'$ we allow the deduction of $C - \{t = t'\}\sigma$ just in case $t\sigma$ and $t'\sigma$ are not unifiable. This is sound because terms that are not unifiable cannot denote the same thing, under any assignment to variables, in models that satisfy our restriction. And it is complete, because two terms can fail to denote the same thing only if the terms are distinct. ∎

Like the axioms of SEq, inference rule (ii) still poses an efficiency problem. The space of possible substitutions that can make a pair of terms distinct is too large to be managed efficiently. We will make some headway on this problem below, but we can note here the logic programmers' strategy. It is just to delay the evaluation of any positive equation (or any other non-ground positive literal) in a clause as long as possible – try to resolve against all the other literals first, in hopes of instantiating the terms in the equations – and then form a resolvent $C - \{t = t'\}$ just in case t and t' are not unifiable.[3] That is, we can modify rule (ii) of our extension to resolution as follows:

(ii') If $(t = t') \in C$ and t and t' are not unifiable, deduce $C - \{t = t'\}$.

[3] Compare, for example, the strategy for soundly implementing negation-as-failure in [Lloyd, 1984], and the more general application of similar strategies on otherwise difficult problems in [Naish, 1985].

This strategy is more tractable, but it is obviously not complete.

4 Skolem functions

With clausal form theorem provers, all problems must be represented in clausal forms: existentially quantified variables must be replaced by Skolem functions. When these are introduced, they must be new to the theory, and so we do not know whether their denotations are distinct from those of other terms. For example, the following is a clausal form of *NN* in which a unary Skolem function $sk1$ has been introduced:

$\{\neg nn(A), A = 0, A = s(sk1(A))\}$ *Theory Cl(NN)*
$\{A \neq 0, nn(A)\}$
$\{\neg nn(A), nn(sk1(A)), A = 0\}$
$\{\neg nn(A), B \neq s(A), nn(B)\}$

The first clause tells us that if $nn(A)$ for some A, then either A is 0 or A is the successor of something in the domain, viz. $sk1(A)$. Obviously, on the intended interpretation of the theory, this function must have numbers as its values, and so we have lost the unique names property. But since the Skolem function is the *only* departure from the unique names restriction, since the theory still contains a set of terms that satisfies the restriction, we can still benefit from a specialized rule of inference that provides standard equality reasoning when necessary, but which takes advantage of the fact that many of the terms must have distinct denotations.

4.1 RUE-NRF resolution

As noted above, there are a number of techniques, such as paramodulation, for building in the standard equality theories for the language of any theory. We adapt a technique defined by Digricoli [Digricoli, 1983] [Digricoli and Harrison, 1986] called *RUE-NRF resolution* that is like paramodulation in building in standard equality, but it builds in the equality theory by extending unification in a way more similar to the e-unification strategies of Bledsoe [Bledsoe and Hines, 1980] and Morris [Morris, 1969]. ("RUE-NRF " stands for "resolution by unification and equality" using the "negative reflexive function".)

In RUE-NRF resolution, if terms t, t' are not unified, they give rise to inequalities in the resolvent. For example, the RUE resolvent of $\{p(a, f(b, c))\} \cup A$ and $\{\neg p(X, f(X, d))\} \cup B$ is $A\sigma \cup B\sigma \cup D$ where σ is a substitution and D is a *disagreement set* for $p(a, f(b, c))\sigma$ and $\neg p(X, f(X, d))\sigma$. For example, when $\sigma = \{X/a\}$, D can be either of the following two disagreement sets: $\{f(b, c) \neq f(a, d)\}$ or $\{b \neq a,\ c \neq d\}$. The former is the "topmost" disagreement set. When resolving on equations, symmetry is captured by obtaining two RUE resolvents. For example, the RUE resolvents of $\{a_1 = a_2\} \cup A$ and $\{b_1 \neq b_2\} \cup B$ are $\{a_1 \neq b_1,\ a_2 \neq b_2\} \cup A \cup B$ and $\{a_2 \neq b_1,\ a_1 \neq b_2\} \cup A \cup B$. A NRF resolvent is essentially the result of RUE resolution with $\{X = X\}$,

which removes an inequality $t \neq t'$ from a clause and introduces a disagreement set for t, t'. The NRF resolvent of $\{f(b,c) \neq f(a,d)\} \cup A$ is $A\sigma \cup D$ where σ is empty and $D = \{b \neq a,\ c \neq d\}$. The formal definition of this intuitive approach is not difficult. Definitions and proofs of the soundness and completeness of RUE-NRF resolution (with factoring) for showing E-unsatisfiability are presented in [Digricoli, 1983] [Digricoli and Harrison, 1986]. (A nice informal account appears in [Stickel , 1986].)

4.2 Making RUE-NRF resolution more efficient

Among the various ways of building the standard equality axioms into an inference rule, the advantage of RUE-NRF resolution for present purposes is that it clearly isolates the problems of restricting the substitutions σ and the disagreement sets D used to define the resolvents without losing completeness. The analogous difficulties with finding restrictive and complete paramodulation strategies show that our options are limited. We consider restrictions on the disagreement sets and restrictions on the substitutions, in turn.

The restrictions on disagreement sets are presented in terms of the following notion:

Definition: A disagreement set D is *viable* in S only if one of the following conditions is satisfied:

(a) D is empty, or

(b) for each term s_i in each equation in D, there is a term t_i in a positive equation in some clause in S such that either s_i unifies with t_i or they have the same principal functor and viable disagreement sets, or

(c) there is a substitution that makes the terms of each inequality in D identical.

Digricoli then considers restrictions on the difference sets that are motivated by the following lemma. We let Eq be the standard equality theory for a theory S.

Definition: An *e-chain* from t to t' in clausal form theory S is a sequence of equations each of which occurs in some clause in S, which is instantiated by some substitution σ,

$$(r_0 = r_1)\sigma,\ (r_1' = r_2)\sigma,\ (r_2' = r_3)\sigma,\ \ldots,\ (r_{k-1} = r_k)\sigma$$

where $k > 0$, $r_0\sigma$ is t, $r_k\sigma$ is t' and for all $0 < i < k$ either $r_i\sigma$ is identical to $r_i'\sigma$ or $r_i\sigma$ and $r_i'\sigma$ have the same principal functor and their respective arguments can be proven equal from $S \cup Eq$.

Lemma 4.1. If t and t' are not identical and $t = t'$ can be proven from $S \cup Eq$, then either

(1) $t = t'$ has the form $f(a_1, \ldots, a_k) = f(b_1, \ldots, b_k)$ where for all $0 < i \leq k$, $a_i = b_i$ can be proven from $S \cup Eq$, or

(2) there is an e-chain from t to t' in S.

Digricoli proves this lemma and uses it to justify two important pruning strategies:

(Viability) Use only viable disagreement sets, and when there is a choice among disagreement sets, always choose the topmost.

(Equality Restriction) RUE resolution of $A \cup \{s_1 = s_2\}$ and $B \cup \{t_1 \neq t_2\}$ is permitted only if at least one pair in $\{\langle s_1, t_1\rangle, \langle s_1, t_2\rangle, \langle s_2, t_1\rangle, \langle s_2, t_2\rangle\}$ is such that its members either unify or have both the same principal functor and a viable disagreement set.

It turns out that the substitutions σ that are used in RUE-NRF resolution can be restricted to most general partial unifiers (MGPUs) with two important exceptions. The MGPU is the substitution obtained from the standard (left-to-right) most general unifier (MGU) algorithm modified so that instead of failing at the first nonunifiable expression, it continues to unify as many constituents as possible. For example, $\{X/a\}$ is the MGPU of $p(X,b)$ and $p(a,c)$. However, this strategy must be qualified as follows:

(RUE subst) In computing the MGPU of $\phi, \neg\psi$ in RUE , always unify an occurrence of a variable X and a corresponding occurrence of term t except in the following case: this occurrence of the variable is the argument of a function $f(\ldots, X, \ldots)$ and there is a viable disagreement set of $\phi, \neg\psi$ containing $f(\ldots, X, \ldots) \neq f(\ldots, t, \ldots)$.

(NRF subst) When applying NRF to a literal $X \neq t$, always unify X and t to erase this literal.

When applying NRF to a literal $f(a_1, \ldots, a_k) \neq f(b_1, \ldots, b_k)$, always unify an occurrence of a variable X and a corresponding occurrence of term t except in the following case: this occurrence of the variable is the argument of an inner function $g(\ldots, X, \ldots)$ which occurs as or in some a_i or b_i $0 < i \leq k$, and there is a viable disagreement set of $f(a_1, \ldots, a_k)$, $f(b_1, \ldots, b_k)$, containing $g(\ldots, X, \ldots) \neq g(\ldots, t, \ldots)$.

Let's call the result of restricting RUE-NRF resolution in the ways Digricoli suggests *restricted RUE-NRF resolution.* Digricoli defends the conjecture that restricted RUE-NRF resolution is a sound and complete method for E-unsatisfiability.

4.3 SEq resolution to exploit syntactic equality

Let $\overline{Sk}$ be a set of terms such that the ground instances of the terms in $\overline{Sk}$ have pairwise distinct denotations. Let Sk be the set of terms not in $\overline{Sk}$. To cover the case where Sk is non-empty, we modify our earlier definition of SEq-satisfiability as follows.

Definition: Let the *axiomatization of the unique names assumption SEq(T) for the language of T relative to the set Sk* be defined to comprise the standard

equality axioms Eq for the language, pairwise inequalities for all the constants and function symbols in $\overline{Sk}$, all instances of † in which τ is in $\overline{Sk}$, and for each function symbol in $\overline{Sk}$ an axiom saying that it is a 1-1 function.

Now, we say that T is *SEq-satisfiable (relative to Sk)* iff $T \cup SEq(T)$ is satisfiable.

We can immediately strengthen the previous lemma in a potentially useful way:

Lemma 4.2. If t and t' are not identical and $t=t'$ can be proven from $S \cup Eq$, then either t, t' are unifiable or at least one of $t, t' \in Sk$.

This follows directly from our definition of Sk. It also follows that there cannot be two elements t, t' in any e-chain that are neither unifiable nor elements of Sk. We can accordingly strengthen condition (b) in the relevant notion of viability, which we will now call "strong viability":

Definition: A disagreement set D is *strongly viable* in S only if one of the following conditions is satisfied:

(a) D is empty, or

(b) no equation in D contains non-unifiable terms neither of which is an element of Sk, and for each term s_i in each equation in D, there is a term t_i in a positive equation in some clause in S such that either s_i unifies with t_i or they have the same principal functor and strongly viable disagreement sets, or

(c) there is a substitution that makes the terms of each inequality in D identical.

We can thus strengthen Digricoli's pruning strategies as follows:

(Strong Viability) Use only strongly viable disagreement sets, and when there is a choice among disagreement sets, always choose the topmost.

(Strong Equality Restriction) RUE resolution of $A \cup \{s_1 = s_2\}$ and $B \cup \{t_1 \neq t_2\}$ is permitted only if at least one pair in $\{\langle s_1, t_1\rangle, \langle s_1, t_2\rangle, \langle s_2, t_1\rangle, \langle s_2, t_2\rangle\}$ is such that its members (i) unify, or (ii) at least one of them is in Sk and they have the same principal functor and a strongly viable disagreement set.

Digricoli conjectures that restricted RUE-NRF resolution is a sound and complete method for E-unsatisfiability.

Theorem 4.2. If restricted RUE-NRF resolution is a sound and complete method for E-unsatisfiability, then so is RUE-NRF resolution with our stronger restrictions.

This result, conditioned on Digricoli's conjecture as it is, is trivial. We have strengthened only the two pruning restrictions, and in each case this involves only minor changes that are straightforwardly justified by the definition of Sk, the definition of "strong viability," and our Lemma 4.2.

We have still built in only the standard axioms of equality and not the unique names axioms $\mathrm{SEq}(\overline{Sk})$ for those terms not in Sk. We cannot use (ii) when Sk is nonempty. There are equations that allow a similar treatment, though. Consider the following relation between terms:

Definition: Terms t and t' are *distinguished* iff either

(a) $t, t' \in \overline{Sk}$ are distinct constants, or

(b) $t, t' \in \overline{Sk}$ have either different functors or distinguished arguments.

We can then use the following analog of (ii) to obtain the effect of all of the pairwise inequalities for terms in $\overline{Sk}$:

(iii) If $(t = t') \in C$, deduce $(C - \{t = t'\})\sigma$ if $t\sigma$ and $t'\sigma$ are distinguished, for any substitution σ

This has exactly the desired effect: an equation of two terms that are not in Sk cannot be true, and hence can be eliminated. We can also get a more tractable incomplete form of this rule, in analogy with (ii'):

(iii') If $(t = t') \in C$, deduce $C - \{t = t'\}$ if t and t' are distinguished.

To cover the case where Sk is nonempty, we must also add a rule which provides the effect of the axioms that say that functions $f \notin Sk$ are 1-1. This is needed because the arguments of these functions can be elements of Sk about which we may need to reason using the standard equality rules of RUE-NRF resolution. For example, with pure syntactic equality $s(sk1) = s(sk2)$ would be refuted using (ii) or (ii'), but if s is not a Skolem function and $sk1$ and $sk2$ are Skolem constants in Sk, we cannot refute $s(sk1) = s(sk2)$ using (iii) or (iii'). In order to be able to deduce $sk1 = sk2$, the following rule suffices – it just has the same effect as the inclusion of the relevant axioms would:

(iv) For any clause C, deduce resolvents of C with any instance of $\{f(X_1, \ldots, X_n) \neq f(Y_1, \ldots, Y_n), X_1 = Y_1, \ldots, X_n = Y_n\}$ for any n-ary function f in $\overline{Sk}$.

We call the strengthened version of RUE-NRF resolution together with (iii) and (iv) *SEq resolution*. SEq resolution gives us the enormous advantage of syntactic equality whenever the set Sk is empty; it gives us RUE-NRF resolution whenever Sk contains all terms; and it gives us appropriate intermediate strategies for the intermediate cases. We have implemented a version of SEq resolution using the incomplete rule (iii') in a model elimination system [Loveland, 1978] [Stickel , 1986].

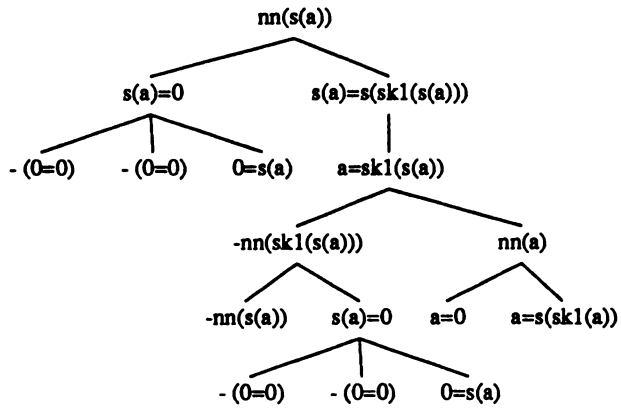


Figure 1: Resolution refutation of $nn(s(a))$

5 Comparing resolution and SEq resolution

Consider the problem suggested in §2. This problem is small enough that we can consider it in detail and display resolution and SEq refutations in a tree format, making some aspects of the comparison graphically obvious. In our refutation trees, each literal immediately dominates the literals in the resolvent of the resolution step it is involved in, and all of the substitutions of the proof have been made throughout the tree. The theory $NN \cup SEq(NN) \cup \{nn(s(a))\}$ is infinite, but we can get a refutation using only the following subset of its clausal form:

$$\begin{array}{ll}
\{\neg nn(A), A=0, A=s(sk1(A))\} & (c12)\\
\{\neg A=0, nn(A)\} & (c13)\\
\{\neg nn(A), nn(sk1(A)), A=0\} & (c14)\\
\{\neg nn(A), \neg B=s(A), nn(B)\} & (c15)\\
\{A=A\} & (c4)\\
\{\neg a=0\} & (c5)\\
\{\neg a=s(A)\} & (c6)\\
\{\neg 0=s(A)\} & (c7)\\
\{s(A)=s(B),\ \neg A=B\} & (c9)\\
\{\neg nn(A),\ \neg A=B,\ nn(B)\} & (c10)\\
\{\neg A=B,\ \neg C=A,\ \neg D=B,\ C=D\} & (c11)
\end{array}$$

Here is a refutation of this theory:

$\{nn(s(a))\}$
clause for refutation
$\{s(a)=0,\ s(a)=s(sk1(s(a)))\}$
using literal 1 *of*(c12)
$\{-A=0,\ -C=A,\ C=s(a),\ s(a)=s(sk1(s(a)))\}$
using literal 3 *of*(c11)
$\{-C=0,\ C=s(a),\ s(a)=s(sk1(s(a)))\}$
using literal 1 *of*(c4)
$\{0=s(a),\ s(a)=s(sk1(s(a)))\}$
using literal 1 *of*(c4)
$\{s(a)=s(sk1(s(a)))\}$
using literal 1 *of*(c7)
$\{a=sk1(s(a))\}$
using literal 1 *of*(c8)
$\{-A=sk1(s(a)),\ -C=A,\ C=a\}$
using literal 3 *of*(c11)
$\{-C=sk1(s(a)),\ C=a\}$
using literal 1 *of*(c4)
$\{sk1(s(a))=a\}$
using literal 1 *of*(c4)
$\{\neg nn(sk1(s(a))),\ nn(a)\}$
using literal 2 *of*(c10)
$\{\neg nn(s(a)),\ s(a)=0,\ nn(a)\}$
using literal 2 *of*(c14)
$\{s(a)=0,\ nn(a)\}$
using clause for refutation
$\{-A=0,\ -C=A,\ C=s(a),\ nn(a)\}$
using literal 3 *of*(c11)
$\{-C=0,\ C=s(a),\ nn(a)\}$
using literal 1 *of*(c4)
$\{0=s(a),\ nn(a)\}$
using literal 1 *of*(c4)
$\{nn(a)\}$
using literal 1 *of*(c7)
$\{a=0,\ s(sk1(a))=0\}$
using literal 1 *of*(c12)
$\{s(sk1(a))=0\}$
using literal 1 *of*(c5)
$\{-A=0,\ -C=A,\ C=s(sk1(a))\}$
using literal 3 *of*(c11)
$\{-C=0,\ C=s(sk1(a))\}$
using literal 1 *of*(c4)
$\{0=s(sk1(a))\}$
using literal 1 *of*(c4)
□
using literal 1 *of*(c7)

This refutation is displayed in Figure 1.

The SEq refutation of the same problem uses only the following theory:

$$\begin{array}{ll}
\{\neg nn(A), A=0, A=s(sk1(A))\} & (c12)\\
\{\neg A=0, nn(A)\} & (c13)\\
\{\neg nn(A), nn(sk1(A)), A=0\} & (c14)\\
\{\neg nn(A), \neg B=s(A), nn(B)\} & (c15)
\end{array}$$

To shorten the proof, it is convenient to ignore the

(RUE subst) restrictions and make use of the following instance of (c12):[4]

$$\{\neg nn(a), a=0, a=s(sk1(a))\} \qquad (c12a)$$

Here is the SEq refutation:

$\{nn(s(a))\}$
clause for refutation
$\{s(a) = 0,\ nn(sk1(s(a)))\}$
using literal 1 *of* (c14)
$\{nn(sk1(s(a)))\}$
using (*iii'*)
$\{\neg sk1(s(a)) = a,\ a = s(sk1(a)),\ a = 0\}$
using literal 1 *of* (c12a)
$\{\neg s(sk1(s(a))) = s(a),\ a = s(sk1(a)),\ a = 0\}$
using (*iv*)
$\{s(a) = 0,\ \neg nn(s(a)),\ a = s(sk1(a)),\ a = 0\}$
using literal 3 *of* (c12)
$\{\neg nn(s(a)),\ a = s(sk1(a)),\ a = 0\}$
using (*iii'*)
$\{a = s(sk1(a)),\ a = 0\}$
using clause for refutation
$\{a = 0\}$
using (*iii'*)
$\square$
using (*iii'*)

This refutation is displayed in Figure 2.

Notice that only one inequality is introduced by RUE-resolution, in the step that uses (c12a). Also notice that in the SEq refutation, the inequalities at the leaves can all be refuted by the relatively efficient but incomplete rule (iii'). The steps of the shorter SEq resolution refutation are not less intuitive than the steps of the previous proof – on the contrary. This derives from the fact that syntactic equality is such an easy idea to grasp, whereas the explicit derivation of syntactic equality results can be tedious and messy. The fact that the SEq refutation is shorter is nice, but unsurprising, given that SEq resolution builds in many axioms. The difference in proof size shown in this small example will clearly be multiplied many times in deductions that involve extensive reasoning with inequalities. Even in a guided search, it is helpful to keep the reasoning about equations as simple as possible, especially when the number of clauses with equations is large.[5] The existence of a large set $\overline{Sk}$ obviously helps simplify equational reasoning.

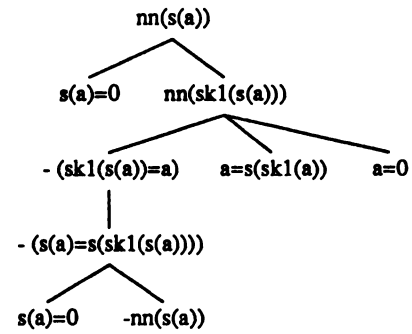


Figure 2: SEq resolution refutation of $nn(s(a))$

The maxims stated in the introduction also make it relevant to compare a problem formulated with syntactic equality vs. the "same" problem formulated without syntactic equality. It is not hard to construct such problems. For example, we can choose to formalize an operation with functions rather than with predicates. Consider the sum function. If we use a sum function symbol in our formalization of arithmetic, then we will have infinitely many different terms denoting each number:

$$s(0) = s(0)+0 = s(0)+0+0 = \ldots$$

If we use a sum predicate, on the other hand, we can respect the syntactic equality restriction, with the exception of necessary Skolem functions and constants. The equations shown above then correspond roughly to a proposition like (the universal closure of) the following:

$$\begin{aligned} & s(0) = X \\ \leftrightarrow\ & sum(s(0), 0, X) \\ \leftrightarrow\ & \exists Y\ \left(sum(s(0), 0, Y) \wedge sum(Y, 0, X)\right) \\ \leftrightarrow\ & \ldots \end{aligned}$$

Comparative studies of alternatives like these have been done before. It is not really surprising that, even when we do not use a proof method that builds in any axioms as SEq resolution does, proofs using a predicate formulation are sometimes less difficult that proofs using a function formulation. For example, in their work on group theory Bellin and Ketonen [Bellin and Ketonen, 1986] point out that, intuitively, a functional representation highlights "intensional features," and consequently many proofs are substantially simpler when

[4] Digricoli shows how any refutation R can be transformed into another refutation R' that satisfies the (RUE subst) and (NRF subst) restrictions in §4.4 of [Digricoli, 1983]. Unfortunately, the transformation is infeasible in general, can dramatically increase the length of the refutation, and has not been shown to be effectively computable in general. The transformation is infeasible because it involves finding a new refutation of a clause not refuted in the original proof, where we know such a refutation exists but not how long it will be.

[5] We have applied these techniques to a complex theory involving equations, as discussed in [Stabler, 1988]. The theory considered there is a formalization of a theory of English syntax in which $\overline{Sk}$ contains all of the non-Skolem constants and function symbols.

we use the "more abstract" predicate representation.[6] In theories of sequences and trees, we have a choice about whether to formalize *append* as a function or predicate, and we have found a predicate formulation respecting syntactic equality to be the most practical in this case [Stabler, 1988].

6 Conclusions

The thesis of this paper is not surprising. The existence of a substantial, decidable set of terms whose denotations are known to be pairwise distinct can be a computational advantage. We have seen how this advantage can be obtained very naturally in one of the approaches that builds in equality and treats unification as a special kind of equation-solving. We extended this approach, RUE-NRF resolution, to SEq resolution, by building in additional axioms that depend on which terms are known to be pairwise distinct. SEq resolution prunes its search for proofs of equations by making use of available information about terms known to have distinct denotations in the intended model.

References

[Bellin and Ketonen, 1986] Gianluigi Bellin and Jussi Ketonen. Experiments in automatic theorem proving. Technical Report No. STAN-CS-87-1155, Department of Computer Science, Stanford University, 1987.

[Bledsoe and Hines, 1980] W.W. Bledsoe and L.M. Hines. Variable elimination and chaining in a resolution-based prover for inequalities. *Procs. 5th Conf. on Automated Deduction:* 70-87, 1980.

[Chang and Lee, 1973] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New York, 1973.

[Clark, 1978] Keith Clark. Negation as failure. In H. Gallaire and J. Minker, *editors. Logic and Data Bases.* Plenum Press, New York, 1978.

[Digricoli and Harrison, 1986] Vincent J. Digricoli and Malcolm C. Harrison. Equality-based binary resolution. *Journal of the ACM,* 33(2): 253–289, 1986.

[Digricoli, 1983] Vincent J. Digricoli. *Resolution by Unification and Equality.* Ph.D. Thesis, Department of Computer Science, New York University, 1983.

[6] The observation comes from a study of proofs done with a proof-checker for second order logic, EKL. Where permutations are treated as bijections of a finite set into itself, Bellin and Ketonen observe that the predicate representation is better for proofs that the identity function is a permutation and that every permutation has an inverse, but the function representation is better for showing that the composition of two permutations is a permutation.

[Flannagan, 1986] Tim Flannagan. The consistency of negation as failure. *Journal of Logic Programming,* 3(2): 93–114, 1986.

[Lloyd, 1984] John W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, New York, 1984.

[Loveland, 1978] Donald W. Loveland. *Automated Theorem Proving: A Logical Basis.* North-Holland, New York, 1978.

[Morris, 1969] J.B. Morris. E-resolution: an extension of resolution to include the equality relation. In *Proceedings of the International Joint Conference on Artificial Intelligence,* pages 287–294, 1969. International Joint Committee on Artificial Intelligence.

[Naish, 1985] Naish, L. (1985) Automating control for logic programs. *Journal of Logic Programming, 3*: 167-183, 1985.

[Reiter, 1965] Ray Reiter. Equality and domain closure in first-order databases. *Journal of the ACM,* 27: 235-249, 1965.

[Robinson, 1965] J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM,* 12: 23–41, 1965.

[Sheperdson, 1984] John C. Sheperdson. Negation as failure: a comparison of Clark's completed data base and Reiter's closed world assumption. *Journal of Logic Programming,* 1(1): 51–79, 1984.

[Stabler, 1988] Edward P. Stabler, Jr. *The Logical Approach to Syntax.* Forthcoming.

[Stickel , 1986] Mark Stickel. A prolog technology theorem prover: implementation by an extended Prolog compiler. In *Procs. 8th Int. Conf. on Automated Deduction,* Lecture Notes in Computer Science Volume 230, pages 573–587. Springer-Verlag, New York, 1986.

[Stickel , 1986] Mark Stickel. An introduction to automated deduction. In Wolfgang Bibel and Philippe Jorrand, *editors. Fundamentals of Artificial Intelligence: An Advanced Course,* Lecture Notes in Computer Science Volume 232, pages 75–132. Springer-Verlag, New York, 1986.

[van Emden and Yukawa , 1980] Maarten van Emden and Keitaro Yukawa. Logic programming with equations. *The Journal of Logic Programming,* 4(4): 265-288, 1980.

[Wos *et al.*, 1980] Larry Wos, Ross Overbeek, and Lawrence Henschen. Hyperparamodulation: A refinement of paramodulation. In *Procs. 5th Conf. on Automated Deduction,* pages 208-219. Springer-Verlag, New York, 1980.

[Wos *et al.*, 1984] Larry Wos, Ross Overbeek, Ewing Lusk, and Jim Boyle. *Automated Reasoning: Introduction and Applications.* Prentice-Hall,Englewood Cliffs, New Jersey, 1984.

Making Situation Calculus Indexical

Devika Subramanian *
Computer Science Department
Cornell University
Ithaca, New York 14850

John Woodfill†
Computer Science Department
Stanford University
Stanford, California 94305

Abstract

In the spirit of Levesque and Brachman[1985], we study a new class of representations made popular in the world of reactive planning [Firby, 1987], called indexical-functionals [Agre and Chapman, 1987]. We present a knowledge level analysis [Newell, 1982] of some indexical terms that clarifies their logical content and helps us identify the source of their computational power. In particular, we demonstrate that indexical terms can be formalized in a restriction of the situation calculus. Indexical theories gain descriptional as well as computational efficiency by using terms whose referents are determined in context by the perceptual machinery of an agent, and by selecting those and only those terms that are essential for determining a course of action for the agent. We show how indexical theories can be synthesized from a situation calculus description of a planning problem together with knowledge of the goals of the agent. This perspective on indexical terms provides us not only with a way of understanding what their ontological underpinnings are, but also helps us analyze the conditions under which they are useful.

1 Introduction

The intractability of classical planning and the need to actively monitor plans in a complex, dynamic world has led to the development of reactive planners that build plans at execution time based solely on the situation existing then [Firby, 1987]. Several designs for reactive planners have been proposed in the recent literature: the theory of situated automata and their combinational circuit compilation by Rosenschein and Kaelbling [1986], the theory of indexical-functional aspects in the reactive planner Pengi proposed by Chapman and Agre, and Rod Brooks' [1987] subsumption architecture. For the purposes of this paper we focus on the indexical-functionals described by Agre and Chapman in [1987] with a view to understanding the assumptions about the world and the planning process they make in order to contain the complexity of planning in their framework. The results we seek are a declarative specification of the worlds in which Pengi style planning works and an analysis of their computational and descriptional efficacy. Our results apply equally well to the other architectures.

The impetus to perform this analysis comes from two sources. One, the existence of the fundamental tradeoff between expressiveness and efficiency in knowledge representation [Levesque and Brachman, 1985] indicates that the computational advantage in indexical-functional representations ought to come from expressiveness limitations. If we can identify them, we can determine the worlds for which they are an epistemologically adequate formalism. Two, we would like to design methods for compiling out indexical-functional aspects from more expressive situation calculus descriptions of the world. Presently the designers of Pengi synthesize these aspects by hand. Should the rules of the world change, they will have to manually reconstruct these indexical-functionals and the situation-action rules that use them. Specifying modifications at the level of individual situation-action rules (or worse circuits) is clumsy and for moderately complex worlds almost impossible. An analysis of this representation in terms of situation calculus allows us to build a compiler for indexical-functional aspects and thus to specify modifications at a reasonable level of description.

The designers of Pengi state that "registering and acting on indexical-function aspects is an alternative to representing and reasoning about complex domains, and avoids combinatorial explosions". We demonstrate that indexical-functionals can be constructed out of the objective ontology of the world assumed by a situation calculus representation using the processes of indexicalization and propositionalization. Theo-

*This author was supported by an IBM fellowship.
†This author was supported in part by ONR Contract N00014-81-K-0004.

ries that contain indexical-functionals simply make different expressiveness-computational tradeoffs as compared to the full situation calculus.

There are many indexical terms in English, "Me", "You", "This", "That", "Here", "There", "Now", "The Car Behind Me" and so on. We begin by introducing time indexicals into situation calculus. A simple example of a time indexical is *Now*. An agent that has its present theory of the world expressed in terms of the logical constant Now alone, is computationally simpler than one that requires expression of statements about time points in terms of some fixed time point other than *Now*. This idea is made rigorous in Section 2. Section 2.1 introduces the axioms needed for meaningful introduction of *Now* in the situation calculus. In Section 3, we use these axioms to reformulate a standard first order predicate logic sentence that assumes an objective ontology into two indexical ones that use *Now*. One of the indexical sentences is tuned for one-step planning, the other is tuned for execution monitoring. The final step is propositionalization that leads to the construction of propositional theories with their well-known efficiency properties.

The introduction of *Now*, sets the stage for introducing further indexicals pertaining to space (*Here* and *There*), and to objects (*The Block-that-is-behind-me*). However, introducing additional indexical terms, necessitates making assumptions about the process of determining their referents. This is the subject of Section 4.1. In Section 4.2 we discuss the expressive limitations of indexicals and present an analysis of the time and space costs associated with using them for monitoring and one-step planning. We conclude with a summary of the main points of the paper in Section 5.

2 A Conceptualization with a View

To use logic one must choose a point of view. Traditional situation calculus starts from a sort of god's eye or objective point of view. This objective stance allows one to frame eternal sentences, e.g. "Block A is on block B at 3:13, May 3rd, 1989". "block A" and "3:13, May 3rd, 1989" are considered as rigid designators that denote the same things for all time. Eternal sentences have the nice property that any conjunction of eternal sentences is also an eternal sentence. If an agent acquires true beliefs in the form of eternal sentences, it can always conjoin these new beliefs to its old true beliefs without fear of contradiction.

It is also possible to develop a situation calculus from an agent's subjective point of view. Consider a subjective ontology. What things and relations are central to an agent's point of view? First, the current time seems interesting, *Now* must be a distinguished thing for an agent. *Tomorrow* and *Yesterday* are important, but less so than *Now*. The place, *Here*, must be something special. If the agent is sitting at its desk typing, the typewriter in front of it, the chair under it, the paperback at its elbow, are all central objects. The agent could have a theory of here-and-now couched in symbols denoting these various elements of the ontology of here-and-now.

As a first example of an indexical theory, let us consider a rather narrow minded agent that makes one distinction, and hence deals with one predicate, *Rich*. *Rich* is true when the agent is rich, and false when the agent is not. Suppose further that the agent owns AI stock that was worth millions yesterday. Yesterday, the agent could have a theory containing the one sentence Rich. Today, the stock market fell and the stock is worth nothing; the agent can add ¬Rich to its supply of facts. The agent seems committed to the sentence: Rich ∧ ¬Rich.

Rich must be made situation dependent. From an agent's point of view, the current moment (not S_0, the first situation) is the most interesting moment, so we introduce a term Now which now denotes the current situation. Our agent can maintain ¬Rich(Now). The agent's state of richness yesterday can be stated today if we introduce a function *Before* which maps one state to its predecessor: Rich(Before(Now)) Notice that Rich-Now is a quite useful propositional symbol where Rich-S0 is not. The question of whether I am rich now is always fairly relevant, while the question of being rich at some fixed time loses significance. In what follows, we will describe how we can convert a theory that contains a situation-dependent predicate like *Rich* expressed in the full situation calculus that allows access to arbitrary situations, into an indexical one that allows reference only to the current situation and its predecessors via the *Before* function.

Two obstacles prevent the use of *Now* and *Before* and their corresponding symbols Now and Before in standard situation calculus: first, *Before* is not definable in standard situation calculus as there is no unique predecessor of a situation; second, it seems that an agent's theory becomes highly non-monotonic across time. Rich(Now) was in the theory yesterday, and it is not today. The first obstacle is overcome by adding appropriate axioms for situations and *Before*. The second obstacle is stepped around by accepting the idea of having a different, current, theory for each new situation in time, and considering the issue of transforming one theory of here-and-now at t_1 to another theory of here-and-now at t_2.

2.1 The Reformulated Situation Calculus

The need to provide more axioms for the situation calculus arises because *Before* is not explicitly definable in the standard situation calculus. To see that *Before* is not definable consider a world containing a compass C and four directions in which the compass' needle can point: N, E, S and W. The two actions in this world, *Clockwise*, and *Anti-Clockwise*, turn the compass. One

model of this world has four situations, one for each direction the needle of the compass can point to.

$$S1 = C(N), S2 = C(E), S3 = C(S), S4 = C(W)$$

If *Before* were definable, $Before(S2)$ would have to be either $S1$ or $S3$. But $Do(Clockwise, S2) = S1$ and $Do(Anti\text{-}Clockwise, S3) = S1$. Clearly there can be no function defined on these four situations which picks out the previous situation.

The following axioms restrict the models of situation calculus to those in which there is a unique previous situation. The variables a1 and a2 range over actions, and s1 and s2 range over situations.

Intuitively, we would like to define a function *Before* which maps a situation into its predecessor, just as the function "the day before" maps today into yesterday. This intuition is captured by the following definition for a relation:

(A). $\{\langle x, y\rangle \mid \exists a\ \mathtt{Do}(a, x) = y\}$

However for many models of situation calculus, this formula is not functional. The definition becomes functional if *Do* satisfies the following sentence.

(B). $\forall y\ \exists a\ x\ \mathtt{Do}(a, x) = y$
$\Rightarrow (\forall z\ \mathtt{Do}(a, z) = y \Rightarrow x = z)$

After restricting *Do* in this way, *Before* is still not defined by (A) for situations which are not in the range of *Do*. We choose to introduce an axiom relating *Before* and *Do* which entails (B) and forces *Before* to agree with (A):

$(ISC1)$. $\forall a\ s\ \mathtt{Before}(\mathtt{Do}(a, s)) = s$

As in our everyday world, given this axiom, each situation has a unique predecessor. However, there may be several distinct actions which take one situation to the same successor. Although not necessary for the construction, in accord with common intuition, we provide an axiom which forces situations with distinct histories to be distinct:

$(ISC2)$. $\forall a1\ a2\ s\ a1 \neq a2$
$\Rightarrow \mathtt{Do}(a1, s) \neq \mathtt{Do}(a2, s)$ *optional*

Later when we discuss transforming one theory of here-and-now to the next theory of here-and-now, we will require that every situation be *before* some situation, (that *Before* be onto) for a soundness condition to hold:

$(ISC3)$. $\forall s1\ \exists s2\ \mathtt{Before}(s2) = s1$

To give an intuition as to what sort of structures these axioms define, we describe their models as labeled directed graphs in which situations are nodes and actions are labels. Considering the case where there is more than one situation and more than one action, a model consists of a set of components as shown in Figure 1. Notice in this model, components have back arcs. Although having back arcs is not inconsistent, we choose to introduce $(ISC4)$ in order to force at least one component to be acyclic.

$(ISC4)$. $\neg\exists a\ s\ \mathtt{Do}(a, s) = S0$ *optional*

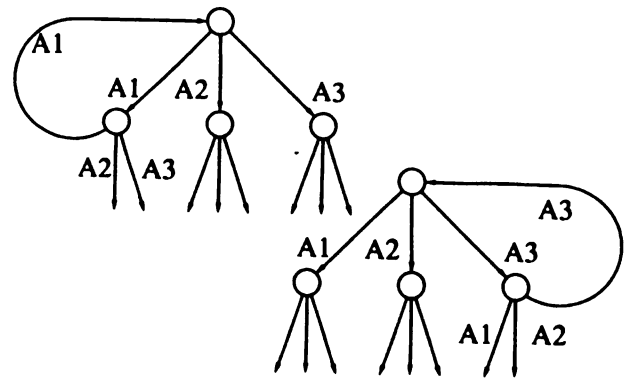


Figure 1: One Model of $ISC1$ - $ISC3$

It picks out an initial situation for a set of standard points, just as 0 picks out an initial element in the standard points of Peano arithmetic.

Given the notion that a situation has a predecessor, the identity of the action resulting in a situation becomes an object of interest[1]. $(ISC5)$ is a definition of an additional relation *Action-that-generated* in terms of *Do*.

$(ISC5)$. $\forall a\ s1\ \mathtt{Action\text{-}that\text{-}generated}(s1, a)$
$\iff \exists s2\ \mathtt{Do}(a, s2) = s1$ *definition*

$\mathtt{Action\text{-}that\text{-}generated}(S, A)$ is true if performing A in some situation would result in S.

A useful consequence of these axioms and definitions is:

$(ISC6)$.
$\forall a\ s\ \mathtt{Action\text{-}that\text{-}generated}(s, a) \iff$
$\mathtt{Do}(a, \mathtt{Before}(s)) = s$ $[ISC1, ISC5]$

2.1.1 Models

To clarify the content of these axioms, we review the structures they define. Again, the structures are described as directed, labeled graphs.

If there is exactly one action, models consist of a set containing one or more component. Components are either cycles or infinite chains.

If there are k actions where $k > 1$ the models differ depending on which optional axioms are included. If $(ISC2)$ and $(ISC4)$ are included, models consist of the standard component, an infinite k-ary tree rooted at $S0$, and zero or more non-standard components which are k-ary, possibly defective, trees. A non-standard component can be defective in the sense that it may include a back arc. Figure 2 depicts such a model where k is 3 and there is one defective component. If $(ISC2)$ is not included, models may have multiple arcs connecting the same pair of nodes in the same

[1] If $(ISC2)$ is omitted, the *identities* of the actions which could have resulted in a situation are objects of interest.

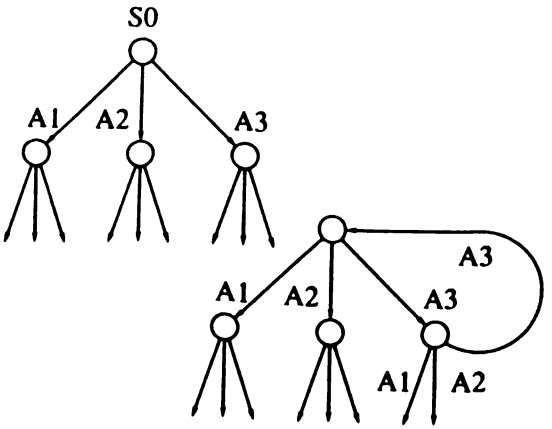


Figure 2: One Model of $ISC1$ - $ISC5$

direction. If ($ISC4$) is not included, some models may have no cycle free component.

All of these are models that have at most one predecessor for each situation, and thus allow the incorporation of *Before*.

2.2 Transforming Theories

At each situation an agent is best served by having the best theory for that situation. In some environments the best theory for a situation is a theory couched in the subjective terms denoting the objects present to the agent in that situation. Each new situation will have a different theory. The theory of one situation is not, in general, a subset of the theory of the succeeding situation. We need a method for generating the current theory in terms of the current subjective ontology that describes the preceding situation. For now, only the term `Now` changes its reference from situation to situation We define a translation which, given a theory of the previous situation, produces a theory that is true in the current situation. The translation maps the term `Now` in the first theory to a term which designates the same thing in the second theory. Since `Before(Now)` now designates the situation which `Now` designated in the previous situation, the translation π maps `Now` to `Before(Now)`, and all other parameters to themselves. To generate the theory of a new situation from the theory of the previous situation one applies the syntactic translation corresponding to π, replacing all occurrences of `Now` by `Before(Now)`. Figure 3 illustrates this idea.

This transformation has the desirable property that it is truth preserving.

Lemma 1 *Given a set of sentences $\mathcal{T}$, and letting ψ denote the sentence* $\forall$`s1` $\exists$`s2 s1 = Before(s2)` *(axiom ($ISC2$) from above),*
and π denote [`Now` $\rightarrow$ `Before(Now)`], *(the replacement of* `Before(Now)` *for* `Now`*),*
for any sentence ϕ:

$$\mathcal{T} \cup \{\psi\} \models \phi$$

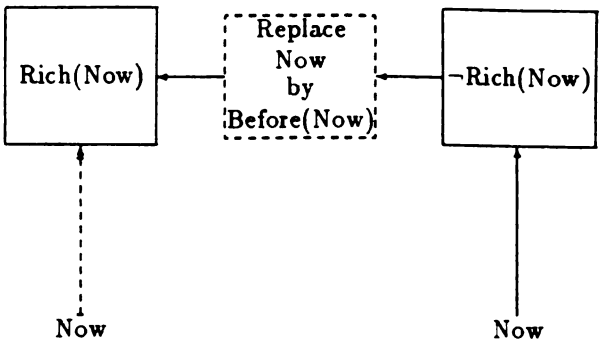


Figure 3: Transforming Theories of the Past

if and only if

$$\pi(\mathcal{T}) \cup \{\psi\} \models \pi(\phi)$$

This simply says that if *Before* is onto, any sentence ϕ is a logical consequence of a theory $\mathcal{T}$ if and only if the result of replacing all occurrences of `Now` in ϕ by `Before(Now)` is a logical consequence of the theory generated by replacing all occurrences of `Now` in $\mathcal{T}$ by `Before(Now)`, that is the transformation preserves the consequences of a theory modulo the shift in designation for `Now`.

Notice that some sentences may be ones in which the term `Now` has been substituted for a universally quantified situation variable. For such sentences it is neither necessary nor useful to apply the transformation. The same sentence about *Now* will be useful in all situations.

3 Reformulations

Now that we have the apparatus for talking about *Now* and *Before*, we consider formulations which are particularly well suited for one-step planning and plan monitoring. In both cases we start with a blocks world axiom from [Genesereth and Nilsson, 1987]:

$$\forall \texttt{s x y} \quad \texttt{T(On(x,y),s)} \wedge \texttt{T(Clear(x),s)} \Rightarrow \texttt{T(Clear(y),Do(U(x,y),s))} \qquad (1)$$

3.1 One-step Planning

For one-step planning, we also need some notion of the relation between goals and what an agent ought to do now. We introduce an axiom of rationality.

($ISC7$). $\forall$`a p s T(p,Do(a,Now))` $\wedge$ `Goal(p,Now)` $\wedge\neg$`T(p,Now)` $\Rightarrow$ `Must(Now) = a`

Which just says that if doing a would result in p, if one has the goal p, and if p doesn't hold now then one must do a.

In one step planning the *Goal* and properties of *Now* are the relevant aspects of the situation, and so the axiom should be reformulated in those terms.

Resolving (ISC7) and (1), so that we are talking about goals,

$$\begin{aligned}\forall x\ y \quad & \mathtt{T(On(x,y),Now)} \wedge \mathtt{T(Clear(x),Now)} \\ & \wedge \neg\mathtt{T(Clear(y),Now)} \wedge \mathtt{Goal(Clear(y),Now)} \\ & \Rightarrow \mathtt{Must(Now)} = \mathtt{U(x,y)} \qquad (2)\end{aligned}$$

Introducing the time indexical Now has propositionalized the situation component of the planning axiom. We can carry this one step further, by instantiating the variables x and y with particular blocks.

$$\begin{aligned}& \mathtt{T(On(A,B),Now)} \wedge \mathtt{T(Clear(A),Now)} \qquad (3) \\ & \wedge \neg\mathtt{T(Clear(B),Now)} \wedge \mathtt{Goal(Clear(B),Now)} \\ & \Rightarrow \mathtt{Must(Now)} = \mathtt{U(A,B)}\end{aligned}$$

And finally introducing propositional symbols to make the sentence propositional,

$$\begin{aligned}& \mathtt{On\text{-}A\text{-}B} \wedge \mathtt{Clear\text{-}A} \\ & \wedge \neg\mathtt{Clear\text{-}B} \wedge \mathtt{Goal\text{-}Clear\text{-}B} \\ & \Rightarrow \mathtt{Must\text{-}U\text{-}A\text{-}B} \qquad (4)\end{aligned}$$

Formula (4) is good for one-step planning because determining the truth value of a set of literals about *Now* determines what the agent must do.

3.2 Plan Monitoring

In plan monitoring, the agent considers what was true in the previous situation, what action was taken, and what should be true now. Thus the relevant temporal terms are Now, and Before(Now).

Again we start with the blocks world axiom (1):

$$\begin{aligned}\forall s\ x\ y \quad & \mathtt{T(On(x,y),s)} \wedge \mathtt{T(Clear(x),s)} \\ & \Rightarrow \mathtt{T(Clear(y),Do(U(x,y),s))}\end{aligned}$$

Universally instantiating Before(s) for s:

$$\begin{aligned}\forall s\ x\ y \quad & \mathtt{T(On(x,y),Before(s))} \qquad (5) \\ & \wedge \mathtt{T(Clear(x),Before(s))} \\ & \Rightarrow \mathtt{T(Clear(y),Do(U(x,y),Before(s)))}\end{aligned}$$

Applying (*ISC*6):

$$\begin{aligned}\forall s\ x\ y \quad & \mathtt{T(On(x,y),Before(s))} \\ & \wedge \mathtt{T(Clear(x),Before(s))} \\ & \wedge \mathtt{Action\text{-}that\text{-}generated(s,U(x,y))} \\ & \Rightarrow \mathtt{T(Clear(y),s)} \qquad (6)\end{aligned}$$

Universally instantiating Now for s:

$$\begin{aligned}\forall x\ y \quad & \mathtt{T(On(x,y),Before(Now))} \\ & \wedge \mathtt{T(Clear(x),Before(Now))} \\ & \wedge \mathtt{Action\text{-}that\text{-}generated(Now,U(x,y))} \\ & \Rightarrow \mathtt{T(Clear(y),Now)} \qquad (7)\end{aligned}$$

Simplifying once, by choosing particular blocks,

$$\begin{aligned}& \mathtt{T(On(A,B),Before(Now))} \\ & \wedge \mathtt{T(Clear(A),Before(Now))} \\ & \wedge \mathtt{Action\text{-}that\text{-}generated(Now,U(A,B))} \\ & \Rightarrow \mathtt{T(Clear(B),Now)} \qquad (8)\end{aligned}$$

Simplifying again by introducing new propositional symbols:

$$\begin{aligned}& \mathtt{On\text{-}A\text{-}B\text{-}Before\text{-}Now} \wedge \mathtt{Clear\text{-}A\text{-}Before\text{-}Now} \\ & \wedge \mathtt{Action\text{-}that\text{-}generated\text{-}Is\text{-}U\text{-}A\text{-}B} \\ & \Rightarrow \mathtt{Clear\text{-}B\text{-}Now} \qquad (9)\end{aligned}$$

Formula (9) is good for monitoring plan execution, since it determines what ought to be the case now in terms of what was true before.

4 Analysis

4.1 Reference

The introduction of indexicals raises the issue of how a term refers to an object in the world. According to traditional Tarskian semantics On(A, B) is true if and only if *A* (the block designated by A) is *On* (the relation designated by On) *B* (the block designated by B). This is fair and good since A, B, and On are considered to be rigid designators that always pick out the blocks *A* and *B*, and the relation *On*.

Another kind of denoting expression, the definite description, has been in disfavor for most of this century. An example of a definite description in English is "The mugger behind the door". A direct translation into logic using the similarly unpopular variable binding operator ι is ιm Mugger(m) $\wedge$ Behind(m, Door). It is easy to see why this form of definite description is annoying. Suppose we define a new constant symbol M to stand for the mugger mentioned above. We can reason as follows: either M is fat or M is not fat. If M is fat then he can take my money because of his weight advantage. If M is not fat then he can take my money by fighting skill. Therefore I'd better run. But this conclusion is silly since there is no mugger behind the door.

Russell[1905] argued that this kind of expression should be translated into logic only in the form of a proposition such as "the mugger behind the door is fat" or

$$\begin{aligned}\exists m \quad & \mathtt{Mugger(m)} \wedge \mathtt{Behind(m,Door)} \\ & \wedge (\forall n\ \mathtt{Mugger(n)} \wedge \mathtt{Behind(n,Door)} \\ & \Rightarrow \mathtt{m} = \mathtt{n}) \Rightarrow \mathtt{Fat(m)}\end{aligned}$$

Unfortunately, agents cannot subsist on rigid designators and suitably guarded definite descriptions. Consider an agent that must determine whether Must(Now) = U(A, B) holds using formula (3):

$$\begin{aligned}& \mathtt{T(On(A,B),Now)} \wedge \mathtt{T(Clear(A),Now)} \\ & \wedge \neg\mathtt{T(Clear(B),Now)} \wedge \mathtt{Goal(Clear(B),Now)} \\ & \Rightarrow \mathtt{Must(Now)} = \mathtt{U(A,B)}\end{aligned}$$

Presumably the agent has $Clear(B)$ as a goal, so the fourth literal can be eliminated. It must now look out into the cruel world. Suppose its builders have been benevolent and written large A's and B's on each face of each block. Then as long as the agent can see one block with A written on it and one block with B written on it, it can determine the truth of the first two literals, and can tell what it must do. But suppose the writing on one block is illegible, there are two blocks with A's on them, there are no blocks with A's on them, or there is a block with both A and B on it... The point is that for an agent in the world there are no sure-fire rigid designators.

We could follow Russell's suggestion and introduce A and B as unary relations, producing the following:

$$\begin{aligned}\exists \mathtt{a\ b} \quad & \mathtt{A(a)} \wedge (\forall \mathtt{a1}\ \mathtt{A(a1)} \Rightarrow \mathtt{a} = \mathtt{a1}) \\ & \wedge \mathtt{B(b)} \wedge (\forall \mathtt{b1}\ \mathtt{B(b1)} \Rightarrow \mathtt{b} = \mathtt{b1}) \\ & \Rightarrow (\mathtt{T(On(a,b),Now)} \wedge \mathtt{T(Clear(a),Now)} \\ & \wedge \neg \mathtt{T(Clear(b),Now)} \wedge \mathtt{Goal(Clear(b),Now)} \\ & \Rightarrow \mathtt{Must(Now)} = \mathtt{U(a,b)})\end{aligned} \tag{10}$$

But who is fooling whom? How is our poor agent supposed to act on this sentence?

When formulating theories for agents in the world, we must accept the problems of failed and multiple reference. Thus A will designate the block that the agent takes to be A. But once we have accepted non-rigid designators and rejected Russell's definite descriptions, we can easily admit such terms as "the-bee-that-is-chasing-me", or more formally ιb Bee(b) $\wedge$ Chasing-me(b).

Notice that there is still major complexity in "perception". To determine if Red(ιb Bee(b) $\wedge$ Chasing-me(b)), the agent's perception must work out which is the bee that is chasing me and what its color is.

4.2 Expressiveness and Computational Efficiency

The class of theories in the situation calculus that we can transform to the indexical form is restricted by the requirement that the theory be consistent with $ISC1$-$ISC4$. This restriction rules out theories that fix the number of situations or that make two situations with distinct histories equivalent.

Claim 1 *A theory $\mathcal{T}$ can be indexicalized if $\mathcal{T} \cup \{ISC1\text{-}ISC4\}$ is consistent.*

Sections 3.1 and 3.2 use the axioms of the indexical situation calculus to generate propositional theories that are suited for plan monitoring and 1-step planning respectively from a theory in the situation calculus. The previous theorem establishes the class of theories for which this conversion is possible. We now show that the compilation methods we use are sound.

Let I be a transformation performed on a theory $\mathcal{T}$ in the situation calculus that obeys the limitation in Claim 1. I is sound if

$$\textit{Soundness.}\quad \mathcal{T} \cup \mathtt{ISC1\text{-}ISC6} \models \mathtt{I}(\mathcal{T})$$

.

We further require that the theory $I(\mathcal{T})$ be propositional, because we wish to gain the computational advantages that reasoning within the propositional fragment of logic gives us.

Claim 2 *The transformations in Sections 3.1 and 3.2 are sound.*

Proof: Follows from the fact that the transformation I in both cases is deduction. □

We examine the computational consequences of indexicalization and propositionalization. First we consider space requirements. To propositionalize a theory $\mathcal{T}$ in clausal form in a typed logic, one must augment the language until there are object constants for each object in each type. The resulting theory $\mathcal{T}'$ is the set of all ground instances of all clauses in the original theory. The cardinality of this set of clauses is expressed by the following formula.

$$|\mathcal{T}'| = \sum_{c \in \mathcal{T}} \prod_{v \in Variables(c)} |type(v)|$$

Considering one type, $\mathcal{D}$, a clause replicates exponentially in the number of variables occurring in it.

$$|\mathcal{T}'| = \sum_{c \in \mathcal{T}} |\mathcal{D}|^{|Variables(c)|}$$

Even a theory with one clause containing one literal with one situation variable in propositional form would contain infinitely many clauses.

Here is where the power of the indexical approach lies. If the agent can do all its reasoning with a few situation terms such as Now and Before(Now), and a few world terms This-Block and That-Block, this explosion may not be significant. Essentially we reduce the size of the theory to

$$|\mathcal{T}'| = \sum_{c \in \mathcal{T}} \prod_{v \in Variables(c)} |type(v)|$$

where type(v) = 1 (Now) when v is a situation variable and type(v) = n (the number of propositional object referents, like the-bee-that-is-chasing-me) when v is an object variable.

Claim 3 *Indexicalization is a win in terms of space only if the number of situation referents and object referents (indexical-functional aspects) is much smaller than the total number of situations and objects described in the theory $\mathcal{T}$ in the situation calculus.*

Suppose we have a theory of cars in the United States expressed in situation calculus. The straightforward propositionalization of this theory would result

in another theory whose size is proportional to 200 million (number of cars in the United States). This theory allows us to name every car. If however, we restricted our attention to cars of interest to the agent: *The-car-behind-me*, *The-car-passing-me* etc.., we would have a propositional theory whose size is proportional to the number of such referents. The power of using these indexical terms is that it gives us implicit quantification: the actual referent of the `The-car-behind-me` is determined by the perceptual machinery of the agent. These indexical terms allow access to that part of the immense propositional theory that is actually needed by the agent.

If the size of the theory does not explode severely (which can be ensured by limiting the number of indexical terms), indexing becomes much more straightforward and efficient. The introduction of propositional constants for objects and situations ensures that unification can be done in constant time.

Claim 4 *The worst case time complexity of reasoning in the indexicalized theory is exponential in the size of that theory.*

However, a really significant performance improvement can be achieved by transforming the propositional theory into a set of equations. This can be done if the propositional theory can be "directionalized". We first subdivide the literals in the theory into the perceptions or the givens (examples are `On-A-B-Before-Now` and `On-A-B-Now`) and the actions or the conclusions (e.g. `Clear-B-Now` and `Must-Unstack-A-B`). To facilitate the directionalization of the reasoning chains in this theory from perceptions to actions, we will assume that the conclusion literals occur in unnegated form.[2] Now, for each conclusion, we collect all minimal conjunctions of the givens which imply it, disjoin the set of conjunctions, and construct a definition of the form *Conclusion* $\equiv$ *Disjunction of Conjunctions.*

Claim 5 *The worst case complexity of concluding the truth of the action literals in the directionalized theory constructed above is linear in their number.*

Proof: Since the truth assignments to the givens can be determined in constant time by the perceptual equipment, the complexity of concluding each action literal in the constructed theory is also a constant. If there are n action literals, we need $O(n)$ time to determine their truths.

The definitions for the action literals can be straightforwardly compiled into combinational circuits (AND gates for conjunctions and OR gates for disjunctions), then the time complexity for concluding *all* the action literals goes to O(1).

[2] This is to avoid problems that arise from negation as failure.

Claim 6 *By compiling the definitions of the action literals into combinational circuits, we can determine their truths in constant time.*

Note that the two results above depend on the fact that perception can be done in constant time. This is actually an assumption that Pengi makes.

Another interesting property of indexicalized theories is that they allow simple descriptions of current situations. The description of the current situation is `Now` in the propositional indexical theory and `Do(an (Do(.... (Do(a1, s0)))))` in the original theory expressed in situation calculus (`s0` is the initial situation). Situations in the past can be accessed by the `Before` function in the indexical theory, so that situations far back in the past are rather clumsy to express; their length is linear in their distance from `Now`. In the first-order theory expressed in situation calculus, the length of a term describing a situation is linear in its distance from the initial situation `s0`.

5 Conclusions

This paper provides a principled account of several forms of indexicality. The introduction of *Now* is a necessary condition for introducing other indexical terms non-rigidly referring to objects. We present a method for generating indexical theories from a domain description in situation calculus. Determining the truth of literals involving indexicals requires that perception be able to determine their referents. Indexical theories can be made into propositional ones with good computational properties. This benefit comes at the cost of losing the ability to rigidly refer to arbitrary objects and situations.

We have shown that indexical-functionals can be explained in terms of a well understood formalism: the situation calculus. In so doing we have exposed some of the expressiveness and efficiency trade-offs resulting from using them.

Acknowledgements

We would like to thank Andy Baker, Michael Genesereth, Jane Hsu, Nils Nilsson, and Stan Rosenschein for helpful discussions.

References

[Agre and Chapman, 1987] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *AAAI*, 1987.

[Brooks, 1987] Rod Brooks. Intelligence without representation. In *Workshop on the Foundations of Artificial Intelligence*, 1987.

[Firby, 1987] R. James Firby. An investigation into reactive planning in complex domains. In *AAAI*, 1987.

[Genesereth and Nilsson, 1987] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, 1987.

[Levesque and Brachman, 1985] Hector J. Levesque and Ronald J. Brachman. A fundamental tradeoff in knowledge representation and reasoning (revised version). In *Readings In Knowledge Representation.* Morgan Kaufmann, 1985.

[Newell, 1982] Alan Newell. The knowledge level. *AIJ*, 18(1), 1982.

[Rosenschein and Kaelbling, 1986] S. J. Rosenschein and L. P. Kaelbling. The synthesis of machines with provably epistemic properties. In *Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge*, 1986.

[Russell, 1905] Bertrand Russell. On denoting. *Mind*, 1905.

Inheritance in Automated Planning

Josh D. Tenenberg*
University of Rochester
Computer Science Department
Rochester, New York 14627
josh@cs.rochester.edu

Abstract

The use of abstraction in planning is explored in order to simplify the reasoning task of an automated agent. An extension of inheritance (ISA) hierarchies from object classes to arbitrary object relations and to the actions of an agent serves to partition a planning system into distinct levels. The problem of maintaining the truth values of assertions at different levels of representation is addressed by stating the precise relationship between the different levels. This gives rise to the following correspondence between solutions: for each abstract solution, there exist *isomorphic* specializations at each lower level. Thus, the presence of abstract solutions strongly constrains the size of the original search space.

1 Introduction

In order to bring about desired states of the world, agents must compose actions in a purposeful fashion over a sustained duration of time. In artificial intelligence, the traditional view of planning is as a mental model of activity, where the agent's actions are represented as operations upon data structures that encode world state. Agents can thus search through representations of world states for those that satisfy their goals, rather than the costlier operation of searching within the real world itself. Unfortunately, these mental search spaces typically grow as an exponent of plan length. To combat this problem, researchers in planning have attempted to encode problems at different levels of abstraction [Alterman, 1987, Kautz, 1987, Knoblock, 1988, Nau, 1987, Sacerdoti, 1974, Yang and Nau, 1988] Few such researchers have dealt effectively with the resulting problems of constructing and maintaining several coherent views of the world, and of specifying the relationship between these different views. This is especially difficult given that planning systems manage propositions whose truth values change over time. This paper summarizes research which formalizes one particular type of abstraction, and explores some of the properties and consequences of its use. The abstraction is a generalization of inheritance (ISA) hierarchies [Brachman, 1979, Hendrix, 1979]. The novelty of this approach emerges from two main sources: 1) the use of inheritance orthogonally throughout a planning system, from inheritance of object types, to relations between object types, to actions that effect object types; 2) the precise specification of the relation between levels, which entails the relation holding between solutions derived at these different levels.

Abstract representations typically differ from lower level representations by distinguishing between those aspects of a domain which can be considered details, and those of primary importance. In the described research, the grouping of object classes into superclasses is used as the basis for making this distinction. An abstract class is characterized by the features common to all of its members; details are taken to be those features that distinguish one subclass from another. For example, `Bottles` and `Cups` can be considered abstractly as `Containers`. Common features include the ability of both to hold liquid and to be poured from; distinguishing features include that `Bottles` have narrow necks and `Cups` have wide mouths. Object classes are used as the basis for an inheritance structuring on relations between objects, and actions applied to objects. Therefore, abstract actions effect relations between elements of abstract object classes. For example, one might abstract the predicate `InBottle(WineX, BottleY)` to `InContainer(LiquidX, ContainerY)`, and abstract the operator `pourBottle(BottleY)` to `pourContainer(ContainerY)`. Abstract plans are specialized by choosing, for each abstract step, a concrete step that achieves the desired effect over a smaller class of objects.

*This work was supported in part by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under Contract Number F30602-85-C-0008 which supports the Northeast Artificial Intelligence Consortium (NAIC).

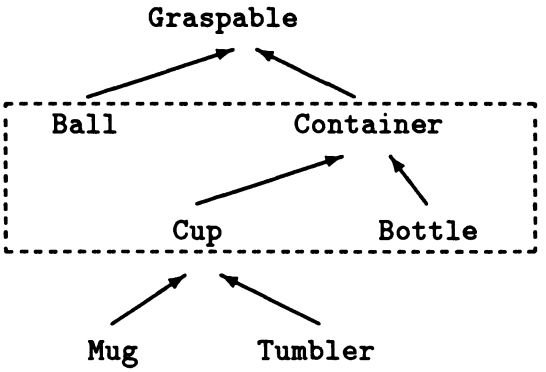


Figure 1: Inheritance Hierarchy

2 Abstracting First-Order Theories

2.1 Predicate Mappings

An extended STRIPS-style language [Fikes and Nilsson, 1971] is used as the base representation. Actions are viewed as operations on sets of sentences denoting world states. Before considering the abstraction of actions, the abstraction of these world states will first be explored. All theorems are stated without proof; such proofs can be found in [Tenenberg, 1988].

Figure 1 shows a fragment of a standard ISA-hierarchy (ignoring for the moment the dotted box). The nodes denote sets of objects, and the arcs denote the subset relation. This relation gives rise to inheritance, for if some property P is true of all elements of a set named by node Q, then by definition, P will be true of all elements of each descendant of Q. For example, anything true of all **Containers** is true of all **Bottles**.

In the described research, the specification of which object classes are grouped together (e.g., **Bottles** and **Cups**) is made meta-linguistically by using a *predicate-mapping function* which renames predicates between a concrete and an abstract level first-order language. Placing this function at the meta-level rather than embedding it implicitly at the object level allows comparison between different abstract class/subclass choices.

No claims are made that, for instance, **Bottles** are an inherently concrete level concept, and **Containers** are inherently abstract, merely that they are concrete and abstract with respect to one another. As pictured in Figure 1, inheritance hierarchies can contain several levels. The formalism that will be provided focuses on only two levels (termed the *concrete* and *abstract*), such as that in the dotted box, but is trivially extendible to arbitrary levels, as will subsequently be seen.

Formally, predicate mappings are functions that map predicate symbols from one first-order language to another. Given two sets of predicate symbols R and R', $f : R \overset{\text{onto}}{\mapsto} R'$ is a predicate mapping, where f is not necessarily $1-1$. This is extended to a mapping between two first-order languages, $f : L \overset{\text{onto}}{\mapsto} L'$, where the predicates are the only symbols that possibly distinguish L and L', and all non-predicate symbols map to themselves under f. Two or more concrete predicates mapping to the same abstract predicate will be called *analogues*. We interpret $f^{-1}(\psi)$ in the obvious way – as the *set* of concrete symbols that maps to ψ under f.

2.2 Model Abstraction

Consider the formal semantic models for two languages[1], L and $f(L) = L'$. Bearing in mind our intuitive notion of inheritance, when might we want to say that a model $\mathcal{M}'$ for L' is an abstraction of a model $\mathcal{M}$ for L?

A reasonable definition is when both models have the same objects in their universes, the interpretations assigned to the non-predicate symbols are identical, and all tuples of the interpretation of an abstract predicate P' in $\mathcal{M}'$ are exactly those tuples of the interpretation of all predicates P in $\mathcal{M}$ that map to P' under f. This is stated formally as follows.

Definition 1 *Let $\mathcal{M} = \langle U, G\rangle$ and $\mathcal{M}' = \langle U', G'\rangle$ be models of languages L and L' respectively, (where U is the universe and G the interpretation function) and $f : L \mapsto L'$ be a predicate mapping function. $\mathcal{M}'$ is* the abstract model of $\mathcal{M}$ through f *(that is, $\mathcal{M}'$ is $\text{AM}_f(\mathcal{M})$) if and only if by definition*

1. $U' = U$,
2. $G'(\psi') = G(\psi')$,
   for all non-predicate symbols $\psi' \in L'$, and
3. $G'(R') = \bigcup_{R \in f^{-1}(R')} G(R)$,
   for all symbols $R' \in \text{Pred}_{L'}$.

Note that neither exceptions nor inductions are captured by the abstraction/specialization relationship between models. If **Cup** maps through f to **Container**, then it is required, without exception, that *every* **Cup** is a **Container** in the corresponding models. In addition, each object taken to be a **Container** must be an element of the extension of a predicate that maps to **Container** (in the corresponding models). No inductions are made that allow objects to be **Containers** that were not one of the known specializations of **Container**.

2.3 Theory Abstraction

The predicate mapping on languages is extended to a mapping on sets of sentences in these languages (which

[1] The symbol $Pred_L$ will be used to denote the predicate symbols of language L.

S:

Bottle(x) ⊃ Graspable(x)
Cup(x) ⊃ Graspable(x)
Bottle(x) ⊃ MadeOfGlass(x)
Cup(x) ⊃ MadeOfCeramic(x)
Bottle(A) ∨ Cup(A)
¬Bottle(B)
¬Cup(B)

f:

f(Bottle) = f(Cup) = Container,
f(Graspable) = Graspable,
f(MadeOfGlass) = Breakable
f(MadeOfCeramic) = Breakable

f(S):

Container(x) ⊃ Graspable(x)
Container(x) ⊃ Breakable(x)
Container(A)
¬Container(B)

Figure 2: Predicate Mapping of Clause Set

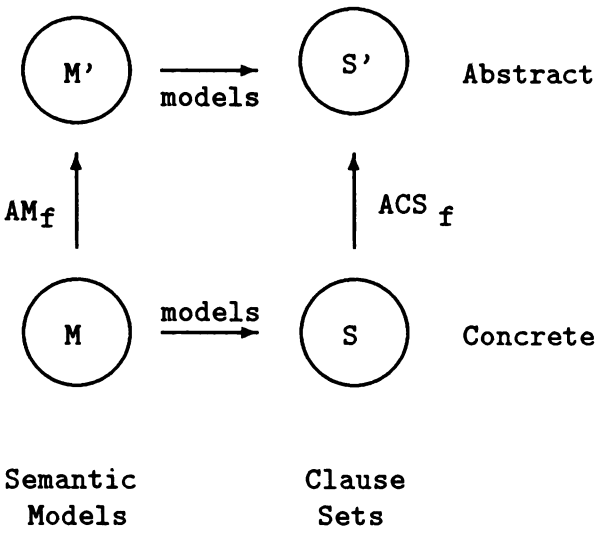


Figure 3: Relationship of ACS_f to AM_f

will be taken to be in clause form[2] [Robinson, 1965]) in the obvious way. Intuitively, this amounts to rewriting the original expression and replacing all predicate symbols by their image under f. By definition, we take $f(\Box) = \Box$, for every abstraction mapping f. An example of a clause set rewritten under a predicate mapping function is provided in Figure 2. Note that $Pred_L$, $Pred'_L$ need not be disjoint.

Having considered the abstraction relation between models, we can similarly consider the abstraction relation between axiomitizations. Given languages L and $f(L) = L'$, when do want to consider a clause set S' in L' to be an abstraction of a clause set S in L? When models for the respective clause sets are in the previously defined abstraction relationship:

Definition 2 *Let S and S' be sets of clauses in L and L', respectively, and let $f : L \mapsto L'$ be a predicate mapping function. S' is an* abstract clause set of S through f *(that is, S' is ACS_f of S) if and only if for every model $\mathcal{M}$ that satisfies S, $\text{AM}_f(\mathcal{M})$ satisfies S'.*

A view of this is provided in Figure 3.

One might reasonably inquire, for a given abstraction mapping f and clause set S, if there exists an S' which is an abstract clause set of S through f? This question is answered affirmatively in the following section, where not only is such an S' shown to exist, but a constructive definition of the strongest such S' is provided.

[2] For simplicity, we will take clauses to be disjunctions of *distinct* literals. That is, no literal will appear more than once in any clause.

2.4 Theory Mappings

Suppose we have an axiomitization in the concrete language that encodes our knowledge about the objects and relations in the world. In addition, we also have a predicate mapping function that indicates the categories of objects and relations that we consider analogous. We would like to construct a weaker axiomitization in the abstract language such that

1. it is faithful to the original axiomitization, in that no statements true in the abstract theory would be falsified at the concrete level,
2. it contains no contradictions, assuming that the concrete axiomitization is consistent,
3. it includes abstract assertions that hold of all specializations,
4. it preserves the abstract model property between the abstract and concrete theories with respect to the predicate mapping, as defined above.

Note that f itself does not satisfy this criteria, since it results in too many clauses at the abstract level, which can result in inconsistency:

$$\texttt{Bottle(A)}, \neg\texttt{Cup(A)} \mapsto_f \texttt{Container(A)}, \neg\texttt{Container(A)}.$$

Definition 4, however, satisfies this criteria. Intuitively, the abstract level never includes axioms whose specializations distinguish between the analogous predicates. Before presenting this definition, provability ($\vdash$) must first be defined.

Definition 3 *A (refutation) proof of C from clause set S is a directed binary tree $T = \langle V, E \rangle$, where* $\Box$

labels the root, leaf nodes are labeled either by elements of S or by $\neg C$ in clause form, and there is an edge $\langle v, w \rangle \in V$ if and only if the label of v is the resolvent under full resolution [Robinson, 1965] from the label of w and some other clause. If there exists a refutation proof of C from S, then C is provable *from S, denoted $S \vdash C$.*

Note that under this definition, $S \vdash C$ if and only if $S \models C$, since full resolution is a sound and refutation complete inference rule. In addition, the same clause can label more than one node of the graph. In particular, this will occur if the clause is used more than once in a proof.

In the following, for any clause C, $|C|$ denotes the number of literals in C, $neg(C)$ denotes the disjunction of the negative literals of C (or $\Box$ if there are none), and $pos(C)$ denotes the disjunction of the positive literals of C (or $\Box$ if there are none).

Definition 4 (Abs Clause Mapping)
$\text{Abs}_f(S) = \{C' |$
for every $N \in f^{-1}(\text{neg}(C'))$ *having* $|\text{neg}(C')|$ *distinct literals,*
there exists $P \in f^{-1}(\text{pos}(C'))$ *such that* $S \vdash N \vee P.\}$

In the degenerate case where C' has no negative literals, the membership condition for C' is not trivially satisfied. Rather, $neg(C')$ is defined as $\Box$, and by definition $f^{-1}(\Box) = \Box$. Therefore, if C' has no negative literals then there exists a unique $N \in f^{-1}(\Box)$ having no literals, namely $\Box$ itself, and it is required that there exist a $P \in f^{-1}(pos(C'))$ such that $S \vdash \Box \vee P$. Put simply, if C' has no negative literals, there must exist a $P \in f^{-1}(pos(C'))$ such that $S \vdash P$. For example,

$$\texttt{Bottle(A)} \vee \texttt{Cup(A)} \mapsto_f \texttt{Container(A)}.$$

The case where there are no positive literals in C' is similar; for every $N \in f^{-1}(neg(C'))$ having $|neg(C')|$ literals, it must be that $S \vdash N$. For example,

$$\neg\texttt{Bottle(B)}, \neg\texttt{Cup(B)} \mapsto_f \neg\texttt{Container(B)}.$$

If C' has neither negative nor positive literals, that is, $C' = \Box$, then $S \vdash \Box$. Therefore, if S is inconsistent, so is $Abs_f(S)$. An example of a clause set mapping under Abs_f is the deductive closure of $f(S)$ from the simple example of Figure 2.

The following theorems and corollaries all hold, proofs of which can be found in [Tenenberg, 1988].

Theorem 1 *S' is* ACS$_f$ *of S if and only if $S' \subseteq \text{Abs}_f(S)$.*

Corollary 2 *If $S' \subseteq \text{Abs}_f(S)$ then S' is inconsistent only if S is inconsistent.*

Corollary 3 *If $\text{Abs}_f(S)$ is consistent then S is consistent.*

Corollary 4 *$DC(\text{Abs}_f(S)) = \text{Abs}_f(S)$, where DC denotes deductive closure.*

Corollary 5 *$\text{Abs}_f(S)$ is finite if and only if S is the empty clause set.*

Corollary 6 *There is no effective procedure for constructing $\text{Abs}_f(S)$, for arbitrary S and f.*

Since every theory S' which is an abstract clause set of S must be a subset of $Abs_f(S)$, Theorem 1 states that Abs_f is as strong as possible. That is, one could not augment $Abs_f(S)$ by adding non-theorems, such that the resulting clause set is ACS$_f$ of S. As a consequence, however, Abs_f is not practical to use, due to the infinite size of the abstract clause sets[3] and the non-existence of an effective procedure for computing it. However, by Theorem 1, since each subset of Abs_f is also ACS$_f$ of S, one can consider subsets of Abs_f that do not have its computational problems but that satisfy the principles given at the beginning of Section 2.4. We demonstrate one such subset below, which additionally has a useful proof-theoretic property.

Definition 5 (MembAbs Clause Mapping)
$\text{MembAbs}_f(S) = \{C' |$
for every $N \in f^{-1}(\text{neg}(C'))$ *having* $|\text{neg}(C')|$ *distinct literals,*
there exists $P \in f^{-1}(\text{pos}(C'))$ *such that* $N \vee P \in S\}$

The only difference between $MembAbs_f$ and Abs_f is that specialized clauses must be *elements of* the original clause set in $MembAbs_f$, whereas they can be *derivable from* the original clause set in Abs_f. This results in the following Lemma:

Lemma 7
If S is a set of atoms, then $\text{MembAbs}_f(S) = f(S)$.

$MembAbs_f$ is computable in the worst case in time quadratic in the size of S, and by Lemma 7, linear in the best case. Note that Corollaries 3, 4, 5, and 6 are not true of $MembAbs_f$.

In addition to the model-theoretic properties associated with $MembAbs_f$ stated above, there is the following important proof-theoretic property. The case where the original clause set, S, is Horn[4] will be stated since it will be required in Section 4, with the more general case of unrestricted clause sets provided in [Tenenberg, 1988].

Theorem 8 *Let S be a Horn clause set in language L, $f : L \mapsto L'$ be a predicate mapping and G' be an atom in L'. If T' is a proof of G' from* $\text{MembAbs}_f(S)$, *then there exists a proof T of G from S such that T is isomorphic to T', and f is a renaming of labels between the nodes in T and their isomorphic images in T'.*

[3] Although it is my (as yet unproven) belief that for finite S, there exists finite S' having the same deductive closure as $Abs_f(S)$.

[4] A *Horn clause* is a clause in which at most one literal is unnegated. A *Horn clause set* is a clause set in which each clause is Horn.

This theorem demonstrates that finding an abstract solution (proof) provides a strong constraint on finding a solution to the original problem, since the only concrete level proofs that need to be pursued are those that exhibit the isomorphism. That is, given an abstract proof, the concrete level search space has been reduced from the set of all proofs to the set of isomorphisms of T'.

3 STRIPS

Thus far, abstraction mappings have been explored with respect to first-order theories. This will be extended to planning systems using a formalization of the STRIPS language, which represents actions as operations on first-order knowledge bases. STRIPS has been used extensively in planning systems [Chapman, 1985, Sacerdoti, 1977, Waldinger, 1977, Wilkins, 1983]. One of the few attempts at a rigorous formalization is provided by [Lifschitz, 1986], from which much of the formalization to be presented has been adapted[5] Although most planning researchers are conversant in STRIPS, I provide a detailed description, since there are several variants described in the literature, with some non-trivial differences between them, and because a crisp formalization allows for a precise statement of the main definitions and theorems. For those familiar with STRIPS, the balance of this section can be skipped, but the following aspects of my definition should be noted:

1. the inferring of *secondary effects* is permitted through the use of a set of *static axioms*, which are elements of each world representation (*situation*), and hence never deleted,
2. there is a suitable description of the set of initial situations that characterize the problem space,
3. all elements of the precondition, add, and delete lists are atomic.

In the original system [Fikes and Nilsson, 1971], STRIPS is both a planning language *and* a stack-based control structure for searching the plan space. In the following, all references to STRIPS refer only to the language component. In STRIPS, the world is viewed as being in a particular state at each moment, and only through the actions of the agent can the world change state. The state of the world is represented by a set of sentences in a first-order language, which will be called a *situation*, and the actions of the agent are represented by *operators*. Associated with each operator is a set of preconditions specifying the conditions that must be satisfied by a situation in order for the operator to apply. The effects of each operator are represented by the deletion and addition of sentences to the situations in which they apply. STRIPS can thus be viewed as a knowledge base (KB) manager, where the effects of an action are realized as operations on the KB. In particular, suppose one has situation S_0 and action o, with the associated sentences A_o to be added, and D_o to be deleted. The new situation resulting from applying o to S_0 is $(S_0 \setminus D_o) \cup A_o$, that is, the old situation minus the deleted sentences, plus the added sentences. By virtue of this syntactic operation, those sentences not deleted continue to hold in the new situation, (the so-called *strips assumption* [Waldinger, 1977]), without the necessity of a separate axiom and inference step for each such sentence, as is typically required in *situation calculus* approaches [McCarthy and Hayes, 1969]. The STRIPS assumption, then, provides a simple approach to handling the *frame problem* [McCarthy and Hayes, 1969].

Since some propositions about the world might inferentially depend upon others, this affects the form in which the world state is encoded so as to facilitate the change in truth value that might occur as the result of applying an action. For instance, suppose that in a typical blocks world scene, there is a stack of blocks. If the top block is removed from the stack, then the fact that it is no longer above the remaining blocks in the stack must be reflected in the axioms used to represent the world. If `Above` is encoded explicitly in the knowledge base, then the operator associated with removing a block must specify the deletion of all of the appropriate `Above` relationships from the KB. Alternatively, one could store only the `On` relationships in the KB, along with an axiom of the form:

$$\forall \mathtt{x},\mathtt{y}.\quad \mathtt{On(x,y)} \supset \mathtt{Above(x,y)}$$

which allows one to infer all of the `Above` relationships. In this way, only the single `On` relation between the top block and the one just below it need be changed as a result of the remove action.

This approach will be generalized to all of the object relationships encoded in the system, and will be reflected in the description of the syntax of STRIPS systems. A predicate will be either *primary*, meaning that it is not dependent upon or derivable from any others, or *secondary*, meaning that it is derivable from the primary relations.

A STRIPS system, Σ is a quintuple (L, E, O, K, σ), where L is a first-order language, E is a subset of the predicates of L (E being the primary predicates), O is a set of operator schemata, K is a non-empty set of clauses in language L (the static axioms), and σ is

[5] The main differences are that I have extended Lifschitz's formalization in 3 significant ways. First, I consider a STRIPS system as defining an entire problem space, not just a single problem instance. Second, I provide a more specific semantics in [Tenenberg, 1988] that associates a set of models with each situation, which provides an interpretation of all of symbols of the logic, not just the truth values of the sentences. And third, I provide a syntactic condition that is necessary and sufficient for the existence of a `Strips`-model for a given STRIPS system.

a set whose elements are sets of ground atoms (the problem space). There are additional constraints on K and σ discussed below.

The set K is taken to be Horn in the balance of this paper. Theorem 9 relies upon the absence of disjunction at the concrete level. Set K includes those clauses which hold in every situation. It is required that every $P \in E$ can occur only in negated literals of clauses in K, that is, as the antecedent of a Horn clause. In essence, this ensures that primary atoms are never derivable in any situation except trivially through membership.

The set E_ϕ is defined as the set of ground atoms formed from predicates in E:

$$E_\phi = \{P(x_1, \ldots, x_n) | P \in E, \text{ and } x_1, \ldots, x_n \text{ are ground terms}\}.$$

Each element of σ is composed from atoms in E_ϕ.

An *operator schema name* is an expression which has the form

$$operator(arg_1, arg_2, \ldots, arg_n)$$

where *operator* is a symbol not in L, and the arg_i are all variables of L. Associated with each operator schema name o is an operator schema *description* (P_o, D_o, A_o), referred to as the preconditions, deletes, and adds, which are sets of atoms. The atoms in D_o and A_o must be atoms from E, which insures that only atomic expressions formed from primary predicates are explicitly managed by the KB operations. The only variables occurring in (P_o, D_o, A_o) are the arg_i of the associated operator schema. An operator schema name together with its description will be called an operator schema. If ϕ is a substitution of terms for variables, and o is an operator schema, then $o\phi$ is understood in the standard way as the substitution of each variable in o by its pairing under ϕ. We take the set O_ϕ to be the set of operator schemata in O under all *ground* substitutions:

$$O_\phi = \{o\phi | o \in O \text{ and } \phi \text{ is a ground substitution}\}.$$

Operator schemata will never occur in expressions partially instantiated. An *operator* is an operator schema fully instantiated by ground terms. Likewise for operator description and operator name. When the context is clear, the term *operator* will often informally be used to refer to just the operator schema name under a particular substitution.

A *dynamic situation* is a subset of E_ϕ, denoting those assertions of a situation that might change truth value as a result of applying an action. A *problem* is a pair $\rho = (T_0, G)$, where T_0 is a dynamic situation called the *initial* dynamic situation, and G, the *goal*, is a sentence of L. Unless specified otherwise, G is taken to be a set (conjunction) of atoms.

A *plan* is a finite sequence of operator names (fully instantiated), with the *length* of the plan being the number of operator names in the sequence. The length n plan $\omega = \langle o_1, \ldots, o_n \rangle$ defines a sequence of dynamic situations $T_0, T_1, \ldots, T_n$, where T_0 is the initial dynamic situation, and

$$T_i = (T_{i-1} \setminus D_{o_i}) \cup A_{o_i}, 1 \leq i \leq n.$$

That is, T_i is the dynamic situation T_{i-1} without the deletes of action o_i but including the adds of action o_i. This in turn defines a sequence of situations $S_0, S_1, \ldots, S_n$ composed of the dynamic situations unioned with the static axioms, i.e.,

$$S_i = T_i \cup K, 0 \leq i \leq n.$$

An operator o is *applicable* in situation S if $S \vdash P_o$. The plan ω is *accepted in* Σ *with respect to* S_0, under the condition that

$$S_i \vdash P_{a_{i+1}}, 1 \leq i < n.$$

where S_i is defined as above. That is, a plan is accepted if each operator is applicable from the situation in which it is applied. S_n, the final situation achieved by executing plan ω from initial situation S_0, is called the *result* and is denoted $Result(\omega, S_0)$. The null plan, denoted $\langle\rangle$, will be considered a plan of length zero accepted in Σ with respect to every situation, where $Result(\langle\rangle, S_0) = S_0$. A situation S_i is *accessible* from S_0 if and only if there exists a plan ω that is accepted in Σ with respect to S_0 and $Result(\omega, S_0) = S_i$. The initial situation is accessible to itself by the existence of the null plan. Plan ω *solves* problem $\rho = (T_0, G)$ if

$$Result(\omega, T_0 \cup K) \vdash G.$$

A problem is *solvable* if there exists a plan that solves it.

The dynamic situations are considered changing parameters of the system. There are only some dynamic situations that we will want to consider as possible values for the changing parameters. These are characterized by the set σ of the quintuple defining Σ. We take σ_K to be the set of elements of σ, each unioned with K:

$$\sigma_K = \{T \cup K | T \in \sigma\}.$$

It is required of all STRIPS systems that σ_K be closed under operator application. That is, for all $S \in \sigma_K$ and all $o \in O_\phi$ applicable in S, $Result(\langle o \rangle, S) \in \sigma_K$. Under this definition, σ_K defines the problem space. That is, all initial situations are drawn from σ_K, and all accessible situations from every $S \in \sigma_K$ are themselves elements of σ_K. For the balance of this paper, the Σ associated with the concrete level a STRIPS system will be taken as defined in Figure 4.

Definition 6 *A* STRIPS *system* $\Sigma=(L, E, O, K, \sigma)$ *is* consistent *if and only if for every situation* $S \in \sigma_K$, *every situation accessible from* S *is (logically) consistent.*

L :

Constants: $\mathtt{C}_1, \mathtt{C}_2, \ldots$
Variables: `w, x, y, z`
Predicates: `On`, `Clear`, $\neq$

E :

`On`, `Clear`

K :

$\neg$`On(TABLE,x)`
$\neg$`On(x,x)`
`On(x,y)` $\supset \neg$`Clear(y)`
`On(x,y)` $\wedge\, y \neq z \supset \neg$`On(x,z)`
`On(x,y)` $\wedge\, y \neq$ `TABLE` $\wedge\, x \neq z \supset \neg$`On(z,y)`
$\{\mathtt{C}_i \neq \mathtt{C}_j | i, j \in \mathbf{N} \text{ and } i \neq j\}$

O :

```
stack(x,y)
  P: On(x,TABLE), Clear(y), Clear(x),
   x ≠ y
  D: On(x,TABLE), Clear(y)
  A: On(x,y)

unstack(x,y)
  P: On(x,y), Clear(x)
  D: On(x,y)
  A: On(x,TABLE), Clear(y)
```

σ :

$\{\mathtt{T} | \mathtt{K} \cup \mathtt{T}$ is consistent and $\mathtt{T} \subseteq E_\phi\}$

Figure 4: Example STRIPS System

A simple example of a STRIPS system for the standard blocks world is given in Figure 4. Note that the static axioms serve primarily as *integrity constraints* [Reiter, 1978]. Since any goal is trivially derivable in an inconsistent situation, one would prefer never to generate such situations through the use of applicable operators. This is particularly important with abstraction, since the abstract solution will be used in constraining the concrete level search. Thus, being misled by inconsistencies to believe that the abstract goal is solved might be excessively costly. There is additionally a semantic motivation for maintaining the consistency of STRIPS systems as one ascends the abstraction hierarchy. A semantics for STRIPS is provided in [Tenenberg, 1988], which demonstrates the existence of a STRIPS-model if and only if the corresponding STRIPS system is consistent.

4 Abstract STRIPS Systems

The intent of abstracting STRIPS systems is similar to that of abstracting first-order theories—to enable the agent to perform search within a smaller space. This requires abstracting all elements of the quintuple that define the STRIPS system, the most difficult of which is the operator set. The intuition associated with abstracting the operators is that actions performed on analogous objects for analogous purposes can be considered the same at the abstract level. These actions will be determined to be analogous based upon a mapping and comparison of the preconditions, deletes, and adds of the actions under the given predicate mapping function.

The STRIPS system resulting from abstracting all operators will have the *downward-solution property*. That is, for every abstract plan ω' solving abstract goal G', there exists some specialization ω of ω', and G of G' such that ω solves G.

Let $\Sigma = (L, E, O, K, \sigma)$ be a STRIPS system and $f : L \mapsto L'$ be a predicate mapping function. The abstract system $\Sigma' = (L', E', O', K', \sigma')$ associated with Σ is itself a STRIPS system, where $K' = MembAbs_f(K)$,

$$\sigma' = \{f(T) | T \in \sigma\},$$

$$E' = \{f(P) | P \in E\},$$

under the restriction that all predicates mapping to the same symbol are either all primary, or all secondary.

The predicate mapping function, f, can be extended to a mapping on operator schemata by assigning to each schema name an abstract name, and mapping each atom of the description to its image under f. For example, the concrete operator `microWave` maps to the abstract operator `cook`, where f maps the predicates in the corresponding positions on the precondition, add, and delete lists:

```
microWave(x,y,z)
  P: In(x,y), InMicroWave(y,z), Raw(x),
  D: Raw(x),
  A: MicroWaved(x)

cook(x,y,z)
  P: In(x,y), InCooker(y,z), Raw(x),
  D: Raw(x),
  A: Cooked(x)
```

Given operator schema $o =$ *opname:* (P, D, A), define

$$f(o) = f(opname) : (f(P), f(D), f(A)).$$

Likewise for instantiated operator schemata. In addition, define

$$f(O) = \{f(o) | o \in O\}.$$

This mapping is extended to plans, so that if $\omega = \langle o_1, \ldots, o_n \rangle$ then $f(\omega) = \langle f(o_1), \ldots, f(o_n) \rangle$.

Only a subset of $f(O)$ will be used in the abstract level system—in particular, those concrete operators that do not distinguish between the analogous predicates. This is similar to how previously, given first-order theory S, only the subset $MembAbs_f(S)$ of $f(S)$ was used for performing abstract level inference. For instance, in order to map `PourFromBottle` to `PourFromContainer` it is required that `PourFromCup` also exist at the concrete level, assuming that cups and bottles are the only predicates that map to container.

Abstracting an operator o from Σ to Σ' requires that there exists in Σ a complete set of analogues of o with respect to the objects over which o is applied.

The set of abstract level operator schemata, $O' \in \Sigma'$, are defined below. It is assumed that all corresponding variables in analogous operator schemata in O have been named identically.

Definition 7 (Operator Abstraction)
$O' = \{o' | o' \in f(O)$ *and there exists* $Q \subseteq O$ *such that*

1. *for each* $o \in Q$, $f(o) = o'$
2. *for each* $P \in f^{-1}(P_{o'})$ *such that each element of* P *is atomic, there exists* $o \in Q$ *such that* $P = P_o$,
3. *for each* $o \in Q$, *for each ground substitution* ϕ, *for each element* $d_i \in D_{o\phi}$, *for each atomic* $d_j \in f^{-1}(f(d_i))$, *either* $d_j \in D_{o\phi}$ *or* $K \cup P_{o\phi} \vdash \neg d_j\}$.

Informally, for each $o' \in O'$ there exists a subset Q of the concrete operator schemata, all of which are analogous and map to o' (under the uniform variable assignments). In addition each specialization of the precondition list of o' is the precondition list for some element of Q. That is, Q is a *complete* set of analogues. And for each element of Q, for every deleted atom, every analogous atom either co-occurs on the delete list, or its negation is derivable whenever the preconditions of this operator hold. This last requirement is imposed in order to insure a correspondence between the abstract and concrete levels, since otherwise, if two analogous propositions hold in some situation at the concrete level and only one is deleted by an operator o, then in the corresponding abstract situation, the abstraction of both propositions will be deleted by the abstraction of o.

A concrete level problem is abstracted as $\rho' = (T', G') = (f(T), f(G))$. Since the abstraction of a STRIPS system is itself a STRIPS system, definitions of legal situation, plan, consistency, etc., hold for the abstract level. Figure 5 illustrates how several operators in a concrete level kitchen domain can map to the same operator at the abstract level (assuming the appropriate predicate mapping function). Note that several different predicates have been abstracted in this example: the cutting surface, the tool, the type of food, and the type of cutting.

5 Downward-Solution Property

Given the abstract STRIPS system defined above, the abstract level will correspond to the concrete level in a precise way. Each abstract plan is specializable by an isomorphic concrete plan such that each intermediate abstract situation is $MembAbs_f$ of the corresponding concrete situation. In addition, each abstract level precondition proof is specializable by an isomorphic concrete level precondition proof for the corresponding operator. Thus, for every abstract level inference, there exists a set of isomorphic images that defines the space of specializations.

Concrete:

```
sliceMushroomWithSlicer(x,y,z)
  P: On(x,z), Countertop(z), Slicer(y),
     Mushroom(x), Held(y), Whole(x)
  D: Whole(x),
  A: Sliced(x)

sliceMushroomWithChefKnife(x,y,z)
  P: On(x,z), CuttingBoard(z), ChefKnife(y),
     Mushroom(x), Held(y), Whole(x)
  D: Whole(x),
  A: Sliced(x)

     ⋮

diceHamWithCleaver(x,y,z)
  P: On(x,z), CuttingBoard(z), Cleaver(y),
     Ham(x), Held(y), Whole(x)
  D: Whole(x),
  A: Diced(x)
```

Abstract:

```
makeInPieces(x,y,z)
  P: On(x,z), CuttingSurface(z), Knife(y),
     Food(x), Held(y), Whole(x)
  D: Whole(x),
  A: InPieces(x)
```

Figure 5: Operator Abstraction

Theorem 9 *Let* $\Sigma'=(L', E', O', K', \sigma')$ *be an abstraction through* f *of the consistent* STRIPS *system* $\Sigma=(L, E, O, K, \sigma)$, *and let* $\omega'=\langle o'_1, \ldots, o'_n\rangle$ *solve* $\rho = (T'_0, G')$ *for some* $T'_0 \in \sigma'$. *For every* $T_0 \in \sigma$ *such that* $f(T_0) = T'_0$, *there exists a plan* $\omega=\langle o_1, \ldots, o_n\rangle$ *accepted at the concrete level such that*

1. $f(\omega) = \omega'$,
2. $\text{MembAbs}_f(\text{Result}(\langle o_1, \ldots, o_m\rangle, T_0 \cup K)) = \text{Result}(\langle o'_1, \ldots, o'_m\rangle, T'_0 \cup K'), 1 \leq m \leq n$,
3. *There exists* $G \in f^{-1}(G')$ *such that*
   $\text{Result}(\omega, T_0 \cup K) \vdash G$,
4. *For each proof* W' *of preconditions* $P_{o'_{m+1}}$ *from*
   $\text{Result}(\langle o'_1, \ldots, o'_m\rangle, T'_0 \cup K')$,
   there exists proof W *of preconditions* $P_{o_{m+1}}$ *from*
   $\text{Result}(\langle o_1, \ldots, o_m\rangle, T_0 \cup K)$
   such that W *and* W' *are related as in Theorem 8,* $0 \leq m < n$.

Figure 6 provides an example of the above theorem under a standard interpretation of the predicates and operators and a suitable predicate mapping.

Note that although the stated theorems are between only two levels, the results extend to additional levels,

Concrete	Abstract
Goal:	*Goal:*
`Diced(A)`	`InPieces(A)`
`Baked(A)`	`Cooked(A)`
Plan:	*Plan:*
`openFridge(F)`	`openFoodSource(F)`
`getFromIn(A,F)`	`getFromIn(A,F)`
`placeOn(A,C)`	`placeOn(A,C)`
`getFronOn(K,C)`	`getFromOn(K,C)`
`sliceHamWKnife(A,K,C)`	`makeInPieces(A,K,C)`
`placeOn(K,C)`	`placeOn(K,C)`
`getFromSurface(A,C)`	`getFromSurface(A,C)`
`putInPot(A,P)`	`putInVessel(A,P)`
`putInMW(P,M)`	`putInCooker(P,M)`
`microWave(A,P,M)`	`cook(A,P,M)`

Figure 6: Plan Abstraction

since the abstract level is itself a STRIPS systems that can be abstracted in precisely the same fashion.

Rather than searching in the original problem space, then, a problem can be abstracted, search can be pursued in the abstract space, and an abstract solution, if found, can be specialized under the constraints of the above correspondence.

Note that the converse of the downward solution property does not hold – there might exist problems that are solvable at the concrete level for which there exists no abstract solution. In particular, these will be problems that rely upon distinguishing features of analogous concrete level classes, for example, a problem requiring the larger size of a conventional oven as opposed to a microwave.

There is, therefore, a delicate balance between the generality of the abstract level and its usefulness. One must trade-off the potential gains of search within increasingly simple spaces against the fewer problems that are solvable within these spaces. I leave it for future work to develop strategies that can *choose* predicate mappings for which this trade-off is optimized.

6 Related Research

There have been several recent suggestions focused upon abstracting actions, such as that of [Nau, 1987, Anderson and Farley, 1988, Alterman, 1987, Kautz, 1987]. All of these researchers propose operators that inherit preconditions and effects, and yet none consider abstract operators to actually *be* operations on an abstract representation of the domain. Therefore, none of their proposals enable the composition of operators, and hence the completion of an entire plan, at the abstract level. Without the ability to compose abstract operators, the meaning of such abstract operators is called into question, as well as the nature of their contribution to the eventual construction of a concrete level plan.

There are several potential benefit of the approach detailed within this paper that enables abstract level plans to be constructed. First, abstract plans allow a form of least commitment, since they do not require specialization of the operations until after the entire abstract level plan has been obtained. For instance, this might allow the specification of which kind of container used in a plan to be pushed closer toward execution time This is reminiscent of Stefik's [1981] constraint posting. Another advantage is that disjunctive information which might stymie a low-level planner need not prevent the formation of an abstract plan. For instance, if object A is either a bottle or a cup, planners that require disambiguation before choosing an action will not be able to proceed, while the planner described above could continue construction of an abstract plan.

7 Future Directions

One type of abstraction that has been considered in other work involves the use of operator composition, typified by MACROPS [Fikes *et al.*, 1972]. In this extension of STRIPS, sequences of operators were parametized by replacing constants by variables, and then stored for possible reuse to solve a subsequent problem. The drawback with this approach is that as the number of stored plans increases, the likelihood that a particular one will apply to a given problem decreases. In addition, the cost of searching the plan library for an appropriate plan grows prohibitively. It was clear to these researchers that there was not sufficient indexical structure to the library:

> The source of this difficulty is that the level of detail in a MACROP description does not match the level of planning at which the MACROP is to be used. ... It may be necessary to consider more sophisticated abstraction schemes involving alteration of predicate meanings or creation of new predicates.

This suggests that it is difficult to exploit the abstraction inherent in composing operators if one has no capability for specifying inheritance types of relationships between the operators. We believe that the predicate abstraction given here provides just such a sophisticated abstraction mechanism.

For instance, in the kitchen-world example, the saving of the specialized plan that was developed might be of little utility, since the same tools and resources might not be available in a subsequent similar problem. In addition, there is a non-trivial overhead cost associated with determining if a saved plan is directly applicable. On the other hand, the abstract plan may

be of sufficient generality so that its frequent use offsets the cost associated with saving it.

Abstraction also appears useful for plan adaptation or repair. That is, if one has a particular plan that is being executed, errors or unforeseen events might occur at runtime–the chef's knife is not available, for instance, or the microwave oven is not functioning. In this case, then, the hierarchical structure of the plan provides a basis for quickly repairing the problem online. This type of computation can be found in [Alterman, 1987] who uses a script-based [Schank and Abelson, 1977] problem representation. When a plan step fails due to the inability to satisfy the precondition of an operator, an analogous operator is attempted that has the same abstract effect, but that has a different specialized precondition. This would amount to using a cleaver as a substitute for the missing chef knife.

8 Conclusion

This research has centered around the principle that in any reasoning system using multiple levels of representation, there should be a precise correspondence between the different levels. In addition, the manner in which inference at one level can guide inference at another level should be made explicit. Such a specification for planning systems has been hindered by the difficulty of managing propositions at two different levels each of whose truth values might change over time. This paper addresses these problems by extending the notion of inheritance from object classes, to relations on object classes, to actions over object classes. A model-theoretic semantics is presented for abstracting first-order theories that generalizes ISA-hierarchies, through the use of a *predicate mapping* function. This mapping is then extended to STRIPS systems. A powerful inferential relationship between levels is shown to hold – abstract plan solutions to problems can be specialized by choosing a specialization of each abstract plan step, and thus concrete solutions that are not isomorphic in this fashion need never be explored.

Acknowledgements

I am indebted to Dana Ballard for his encouragement and critical insights in his role as my dissertation advisor. In addition, I would like to thank Leo Hartman, Jay Weber, James Allen and Lenhart Schubert for the numerous readings of drafts and fruitful discussions.

References

[Alterman, 1987] Richard Alterman. Issues in adaptive planning. Technical Report 304, University of California at Berkeley, 1987.

[Anderson and Farley, 1988] J. Anderson and A. Farley. Plan abstraction based on operator generalization. In *Proceedings of the 7th AAAI*, 1988.

[Brachman, 1979] Ron Brachman. On the epistemological status of semantic networks. In *Associative Networks*. Findler, Nicholas (ed.), Academic Press, 1979.

[Chapman, 1985] David Chapman. Planning for conjunctive goals. AI Technical Report 802, Massachusetts Institute of Technology, 1985.

[Fikes and Nilsson, 1971] Richard Fikes and Nils Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Fikes *et al.*, 1972] Richard Fikes, Peter Hart, and Nils Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[Hendrix, 1979] Gary Hendrix. Encoding knowledge in partitioned networks. In *Associative Networks*. Findler, Nicholas (ed.), Academic Press, 1979.

[Kautz, 1987] Henry Kautz. *A Formal Theory of Plan Recognition*. PhD thesis, University of Rochester, Department of Computer Science, Rochester, New York, 1987.

[Knoblock, 1988] Craig Knoblock. Automatically generating abstractions for planning. In *Proceedings of the First International Workshop in Change of Representation and Inductive Bias*, pages 53–65, 1988.

[Lifschitz, 1986] Vladimir Lifschitz. On the semantics of strips. In *Proceedings of the Workshop on Reasoning about Actions and Plans*, Timberline, Oregon, 1986.

[McCarthy and Hayes, 1969] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*. Edinburgh University Press, 1969.

[Nau, 1987] Dana Nau. Hierarchical abstraction for process planning. In *Proceedings of Second International Conference in Applications of Artificial Intelligence in Engineering*, 1987.

[Reiter, 1978] Raymond Reiter. On closed world data bases. In *Logic and Data Bases*. Gallaire and Minker (eds.), Plenum Publishing Corporation, 1978.

[Robinson, 1965] J. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[Sacerdoti, 1974] Earl Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[Sacerdoti, 1977] Earl Sacerdoti. *A Structure for Plans and Behavior*. American Elsevier, 1977.

[Schank and Abelson, 1977] R. Schank and R. Abelson. *Scripts, Plans, Goals and Understanding.* Lawrence Erlbaum Associates, 1977.

[Stefik, 1981] Mark Stefik. Planning with constraints. *Artificial Intelligence*, 16(2):111–140, 1981.

[Tenenberg, 1988] Josh Tenenberg. *Abstraction in Planning.* PhD thesis, University of Rochester, Dept. of Computer Science, Rochester, NY, May 1988.

[Waldinger, 1977] R. Waldinger. Achieving several goals simultaneously. In *Machine Intelligence 8.* Elcock and Michie (eds.), Ellis Horwood, 1977.

[Wilkins, 1983] David Wilkins. Representation in a domain-independent planner. In *Proceedings of the 8th IJCAI*, 1983.

[Yang and Nau, 1988] Qiang Yang and Dana Nau. The formal specification of task reduction schemas for hierarchical planning. *University of Maryland*, 1988.

Cardinalities and Well Orderings in a Common-Sense Set Theory

Wlodek Zadrozny
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

Abstract

We argue that being a set can be relative to a theory. We introduce the notion of "actually constructed" sets, and we define models of common-sense set theories. We show that certain common-sense intuitions about cardinalities and well-orderings can be expressed in our framework by relativising set-theoretical formulas to the class of "actual sets". We propose a method of talking about sets which permits some sets to be well-ordered without having a cardinality, and vice versa -- to have sets with known number of elements which do not necessarily form a well-ordering. In another application, we explain how certain concepts may involve some natural numbers, but not others.

1. Introduction: motivation

Set theory may be useful in AI, as J. McCarthy suggested in his address at IJCAI-85. Since that time a few books and papers appeared which relate to this topic. Barwise and Etchemendy [1987] used ZFC/AFA set theory to analyze self reference and the liar paradox. Although their work was motivated by technical problems of situation semantics [Barwise and Perry 1983], their elegant results can be interpreted as applications of set theory in AI. Perlis [1987a, 1987b] proposed modifications of ZFC and Ackermann set theories (e.g. adding sets of names of formulas) as a basis for commonsense reasoning. Delgrande [1987] introduced "reducible" and "irreducible" (ones whose members we don't know) sets within a version of ZF, and applied them to an investigation of learning by examples. The proposals for logic programming with sets, like LDL [Beeri *et al.* 1987, Shmueli and Naqvi, 1987] or LPS [Kuper, 1987], express the relevance of set theory for logic programming and database communities (cf. also [Ohsuga and Yamauchi, 1985]). Earlier, Schwartz and his colleagues [Breban *et al.*, 1981, Ferro *et al.*, 1980, Ferro, 1981, Kruchten *et al.*, 1984] proposed a weak set theory as a specification language (SETL).
A common thread of these different approaches is the necessity of modifying the classical set theories in order to deal with specific problems, despite the fact that the classical set theories provide an intellectual framework for thinking about sets.

We too will modify classical set theoretic concepts, but we will not propose a universal set theory for all kinds of reasoning about sets. Rather, we will suggest a way of thinking about cardinalities and well-orderings of finite sets. The two examples below motivate our discussion of common-sense set theory.

Example 1.

Imagine putting into an empty box 75 black and 25 white balls. Clearly, there are 100 balls in the box or -- speaking more formally -- the cardinality of the set of balls is 100. After shaking the box, we would be tempted to say that the balls in the box are not ordered. But this is forbidden by classical set theory, because any set of finite cardinality is supposed to have a well-ordering. (We assume "not ordered" to be a paraphrase of "does not have a well ordering").

Example 2.

We suggest that the reader perform a small experiment consisting of looking for two seconds at the dots at the bottom of the next page. After that the reader should be able to say that this set is (well)

ordered. But almost certainly he does not know the number of dots, although he can assert that there are more than five of them. Again, classical set theory says that if a finite set is well-ordered then it does have a cardinal number which is a natural number.

□

Of course in any of these examples we could distinguish between knowing that something (a cardinality or a well-ordering) exists and being able to point to one. This would allow us to accept the standard set theoretical universes as the "real" sets, at the price of extending the language by adding modalities. Such an approach however does not seem to us to be the most natural solution: introducing modalities usually does not result in clean theories with natural semantics. After all, the balls in the box are not well-ordered; they can be, if we take them out and order them.

We shall propose then a method of talking about sets which permits some sets to be well-ordered without having an exact cardinality, and vice versa - to have sets with known number of elements which do not necessarily form a well-ordering.

The plan of the paper: The next section introduces a few set theoretic concepts needed to understand the main idea of this work. Throughout the paper we will try to help the reader understand the concepts by adding illustrations to the text; their purpose is to describe some of the intuitions we associate with those concepts. A reader with any knowledge of axiomatic set theory can safely omit them. Section 3 defines the set-theoretic framework we will work in (the ***actually constructed sets***). Section 4 redefines the notions of cardinality and well-ordering. In Section 5 we prove that there can be finite sets without well-orderings, as well as finite well-orderings which are not sets. We conclude with a short discussion about reasoning with common-sense sets, and an another example which can be well explained in our theory -- why "I have two children" involves the numbers 2 and 3, but not the number 33323.

2. Basic set theoretical concepts

The purpose of this section is to explain the vocabulary we use. The descriptions of concepts will be rather informal. We first describe some entries in a dictionary of axiomatic set theory (if it were to exist). Then we give a short description of the main domains of set theory in general.

Axiomatic set theory

The theory of Zermelo and Fraenkel, denoted by ZF or ZFC (C standing for the axiom of choice), is the most important axiomatic set theory. In this theory, the existence of a set, such as the set of all natural numbers, is derived from axioms. These axioms are formulated in the standard first order language containing additionally the relation of membership ϵ. For instance, the axiom of infinity says:

$$(\exists x)(\ o \epsilon x\ \&\ (\forall y \epsilon x)(y \cup \{y\} \epsilon x\ ))$$

By rejecting some of the axioms of ZFC, in particular the axiom of infinity, and slightly reformulating the other axioms, we obtain a weaker set theory -- the theory of Kripke and Platek or KPU. In KPU, individuals without elements, called ***points*** or ***atoms***, are allowed to exist (while in ZFC the only set without members is the empty set o). Other concepts to which we are going to refer throughout the paper include:

- ***cardinality*** -- i.e. the number of elements. In standard set theories, elements of finite sets are counted as usual, and counting them is not considered interesting. Counting elements of infinite sets, like all reals, turns out to be a tricky problem; *the continuum hypothesis* is probably the most discussed question of axiomatic set theory. Cantor's paradox arises when the set of all sets, if it were to exist, does not have a cardinality.
- ***hereditarily finite*** -- the set $\{\omega, \omega \cup \{\omega\}\}$ has only two elements, hence is finite. It is not hereditarily finite because the symbol ω traditionally denotes an infinite set -- the set of all natural numbers.
- ***well-ordering*** -- the set of all natural numbers is well-ordered by the relation of $<$ because it is linearly ordered and there is no infinite descending chain of natural numbers $\ldots < x_2 < x_1 < x_0$. Rational numbers or all non-negative reals are not well-ordered by $<$.

- ***natural numbers*** are typically represented by sets: $0 = o$, $1 = \{o\}$, ... , $n + 1 = n \cup \{n\}$, but other representations are also allowed.
- ***models*** of a set theory are collections of sets in which all axioms of this set theory are true. In particular, a model must be closed under the pair formation $\{x, y\}$ and the union $\cup x = \{y : \exists z \in x[y \in z]\}$. The reader may want to check that $V_n[o]$'s are not models of set theory.

Set theory does not allow all possible collections of objects to exist. It describes sets by dividing multiplicities into "consistent" and "inconsistent" ones. Thus $\{X : \neg (X \in X)\}$ (the Russell class of X's which are not members of themselves) is such an inconsistent multiplicity, and its existence is forbidden by the dominant set theories. The discovery of inconsistent multiplicities led to the development of axiomatic set theories.

Set theory was created in the XIX century (by G. Cantor), and axiomatized in the XXth, as a mathematical theory of finite and actually infinite sets. It canonizes the main principles accepted by mathematicians as true about sets. It is remarkable that most of mathematics can be derived from that single source, and that the whole of mathematics can be formalized in it. Together with the results quoted in the introduction this fact would suggest importance of set theoretical concepts for AI. But it is also quite striking that neither "The Oxford Companion to The Mind" [Gregory, 1987] nor the "Handbook of Human Intelligence" [Sternberg, 1982] have "mathematics" (or "set") as an entry. Thus, while set theory is a very powerful tool, the relevance of mathematical reasoning and its foundations to the study of thinking in general -- i.e. to artificial intelligence -- has not yet been demonstrated beyond reasonable doubt.

The fact that all the above mentioned attempts to use set theory in computer science required its modification, the fact that alternatives to the Cantorian set theory are being developed by set theorists [Vopenka, 1979], and the counter-intuitiveness of many theorems of set theory suggest that there can be more than one way of reasoning about sets. We intend to show in the remainder of this paper that there can be many formal ways of reasoning about finite sets and their most elementary properties.

Domains of set theory

The investigation of sets can be divided into several topics:

1. Set algebras (dealing with union, intersection, difference); notice that the membership symbol does not appear in rules like $A \cup (B \cap C) = (A \cap B) \cup (A \cap C)$
2. Theories of cardinality (dealing with counting); typical example would be rules of cardinal arithmetic like $\aleph_5 * \aleph_7 = \aleph_7$.
3. Theories of well-orderings investigating what kind of sets (e.g. of real numbers) can be well-ordered and under which conditions.
4. Descriptive set theory, which provides foundations for the theory of probability and mathematical analysis.
5. Meta-theories of set theory, dealing with questions like which axioms systems are consistent with the existence of the Russellian set of all sets, or with different versions of the continuum hypothesis.

Axiomatic set theories try to settle questions belonging to all these topics, but many of them can be investigated outside of ZFC or KPU. The theme of this paper is related to 2, 3 and 5, but the main distinction between classical set theory and this paper lies in the domain of investigation -- ours is that of finite sets. This approach of formalizing some basic intuitions is not inconsistent with Mac Lane's [1981] appeal for a renewed study of the foundations of mathematics, where mathematics is to be understood as the discovery of formal structures underlying the world and human activities in that world. The following quote may also be relevant in the context of using set theories in AI:

> *The logicism of Frege and Russell tries to reduce mathematics to logic. This seemed to be an excellent program, but when it was put into effect, it turned out that there is simply no logic strong enough to encompass the whole of mathematics. Thus what remained from this program is a reduction of mathematics to set theory. This can hardly be said to be a satisfactory solution of the problem of foundations of mathematics since among all mathematical theories it is just the theory of sets that requires clarification more than any other.*
>
> [Mostowski, 1964]

3. Sets and classes

Sets are built from atoms; classes are collections of sets satisfying certain predicates. We keep these basic premises of classical set theories in our investigation of sets and classes. We will however have two categories of sets: potential -- corresponding to hereditarily finite sets of standard set theories, and actual -- those whose existence is postulated by a particular common-sense theory.

3.1 Language, sets and classes

The theories that we consider are expressed in a standard first order language of set theory (KPU [Barwise, 1985] or ZF [Felgner, 1971, Jech, 1978]). The membership ϵ and two other symbols, # and [...], will be at the center of our attention. The symbol # denotes a cardinality function, while [a , b , c , ...] stands for a well-ordering (an ordered list) consisting of the just mentioned elements. Also, we suppose that among the constants of our language we have an ordered collection **Nums** (which may be disjoint with the atoms) of constants denoting cardinal numerals: **zero (0), one (1) , two (2), ...** . The standard set theoretic symbols are defined as usual. For instance { x } is "the unique y whose only member is x."

The class of Δ_0 - formulas is the smallest class containing the atomic formulas and closed under the connectives and bounded quantification. A formula ϕ is in the class Σ_1 if it has the form

$$\exists y_1 y_2 \cdots y_n \ \psi(y_1 y_2 \cdots y_n)$$

and ψ is Δ_0. The class Σ is the smallest class containing Δ_0 and closed under conjunction, disjunction, bounded quantification and existential quantification (cf. [Barwise, 1975]). It follows from the Σ Reflection Principle (the Theorem below) that Σ formulas are equivalent to Σ_1 formulas (in KPU or ZF).

Illustration 3.1

- The following are examples of Δ_0-formulas:
  - being an unordered pair, $a = \{b, c\}$: $b \epsilon a \,\&\, c \epsilon a \,\&\, \forall z \epsilon a [\, z = b \vee z = c \,]$
  - being an ordered pair, $a = \langle b, c \rangle$: $\exists x \epsilon a \, \exists y \epsilon a \, [\, a = \{x, y\} \,\&\, x = \{b\} \,\&\, y = \{x, c\} \,]$
  - being a transitive set, $Trans(x)$: $\forall y \epsilon x \, \forall z \epsilon y \, [\, z \epsilon x \,]$
- The formula "each element of a belongs to a transitive set", $\forall x \epsilon a \, \exists b \, [\, Trans(b) \,\&\, x \epsilon b \,]$, is in the class Σ but not Σ_1;
- An equivalent Σ_1 formula -- $\exists c \, \forall x \epsilon a \, \exists b \epsilon c \, [\, Trans(b) \,\&\, x \epsilon b \,]$ -- says that the transitive sets b belong to an another set c.

3.2 Potential sets and real sets

We consider objects in a certain domain, some of which are composite (i.e. sets). A theory T refers to these objects directly or indirectly. For instance,

T1. $x \in y \iff x = \text{gold} \vee x = \{ \text{silver}, \text{platinum} \}$

T2. $Nat(y) \iff y = 0 \vee \exists x [\, Nat(x) \,\&\, y = x \cup \{x\} \,]$

T1 refers directly to the atoms 'gold', 'platinum' and 'silver' , and to the set { 'silver' , 'platinum' } ; and indirectly to { gold', { 'silver', 'platinum' }}. T2 refers indirectly to the set of all von Neumann natural numbers greater than 0, and directly to the atom 0.

In both of these cases no reference is made to any other sets; the existence of other sets is not postulated by any of these theories. We believe then that the universe of potential sets, described by a set theory like ZF or KPU, should be separated from the universe of sets actually constructed or referred to by a given common-sense theory. We assume that there exists a collection A of primitive objects (atoms), such that all sets are obtained from atoms by finitely many applications of the singleton operator { } and the union operator $\bigcup$. This determines the universe of potential sets $V_\omega[A]$, where

$$V_0[A] = A,$$

$$V_{n+1}[A] = \text{Powerset}(V_n[A]),$$

$$V_\omega[A] = \bigcup_n V_n[A].$$

It is well known that $\langle V_\omega[A] , \epsilon \rangle$ is a model of KPU set theory. Thus, our universe of potential sets is a natural one and classical. But these sets almost certainly must be treated as platonic (not real) entities: the set $V_6[0]$ has about $2^{64,000}$ elements. It is time then to turn our attention to the actual sets.

But before we can formally define them, we have to write a few paragraphs about classes.

3.3 Classes

Classes are extensions of predicates. For a given theory T, we can consider the collection *Pred(T)* of all predicates appearing in T. If $Pred(P) = (P_1(\ ), \ldots, P_n(\ ))$ then $Classes = (P_1, \ldots, P_n)$, where

$$P_i = \{ (x_1, \ldots, x_n) : V_\omega[A] \models P_i(x_1, \ldots, x_n) \}.$$

We say that a theory T is *unambiguous* if it is satisfiable over $< V_\omega[A], \epsilon >$ and its classes are the same under any interpretation (that makes T true). We assume that all theories refer to finite sets only, and therefore the natural models of them are $< V_\omega[A], \epsilon, Classes >$.

For instance, Pred(T2) = (Nat( )), and the class Nat (which is the only element of the *Classes*) consists of $0, \{0\}, \{0, \{0\}\}, \ldots$ The theory T1 does not describe any class. Both theories are unambiguous. The theory T3 below is an example of an ambiguous theory.

T3. $\forall x[P(x) \rightarrow x = \text{gold} \vee x = \text{silver}]$;
$\exists! x\ P(x)$.

3.4 Sets and Classes

Our analysis of sets and classes for a fixed theory T gives us the structure

$$\mathcal{M}_0(T) = < V_\omega[A], Sets, \epsilon, Classes >,$$

where *Sets* are the sets referred to by the theory T, and *Classes* are the classes corresponding to predicates of T. We have yet to define *Sets* formally. We restrict our attention to unambiguous Σ_1 theories. The restriction is a natural one, since Σ_1 relations on $< V_\omega[A], \epsilon >$ are exactly the recursively enumerable relations (cf. [Barwise, 1975, pp.46-51], for a more precise formulation of this fact). Also we know that

Theorem (The Σ Reflection Principle). *For any Σ formula ϕ,*

$$KPU \vdash \phi \leftrightarrow \exists u\ (\phi)^u.$$

In particular, we can assume that the u is transitive.

(cf. [Barwise, 1975, pp. 15-16]).

Since $(\phi)^A$ is obtained from ϕ by relativization of all quantifiers to A, the theorem says that a Σ_1 formula is true if and only if it is true in some set. This type of fact is called a reflection principle because it describes a situation in which the truth about the big universe (of sets) is locally describable. This principle justifies the equivalence of the two formulas in **Illustration 3.1**.

Let T be a satisfiable (over $< V_\omega[A], \epsilon >$) unambiguous theory. We can now define the referent Ref_Φ of a formula Φ belonging to T:

Definition. Let Ref_Φ be the smallest transitive set A s.t. $(\Phi)^A$ holds, if it exists; otherwise we set Ref_Φ to be $V_n[A]$, where n is a smallest number with the property $(\Phi)^{V_n[A]}$. The class *Refs* of referents of the theory T is described by

$$Refs = \{Ref_\Phi : \Phi \text{ is a formula of T}\}.$$

Note: In the examples we are going to consider that "smallest transitive set A" will always exist. The only role the above theorem plays in this paper is to ensure the correctness of the definition of referents. As we have tried to explain above, the referents are sets whose existence is necessary for building a model of the theory T. Their function is similar to the role of the Herbrand universe for first order formulas.

It is natural to suppose that if a theory refers to a set, it also refers to its elements, elements of its elements, and so on. Therefore

$$Sets = TC(Refs) \cup TC(\cup Classes)$$

The $TC(x)$ is the ***transitive closure*** operator TC applied to the set x, and defined as the smallest transitive set containing x, or as

$$TC(x) = x \cup (\cup x) \cup (\cup\cup x) \cup \ldots.$$

Illustration 3.4:

- For $A = \{\{a\}, \{\{b\}\}\}$, the $TC(A)$ contains a, $\{a\}$, b, $\{b\}$, $\{\{b\}\}$, but not $\{a, b\}$.
- The transitive closure of all odd numbers below 100 will contain all natural numbers smaller than 100.
- The smallest transitive A such that $\phi \leftrightarrow (\phi)^A$ does not have to exist, even when ϕ belongs to an unambiguous theory. Consider $\forall x[p(x) \leftrightarrow [x = \{a\} \vee x = \{b\}]]$ and $\neg p(\{a\})$; this theory is unambiguous but there is no

smallest transitive set relativising the first formula; thus its referent is $V_2[A]$ (assuming a, b are atoms).
A counting function for any set a is Σ_1, thus can be relativised, for instance to the smallest $V_n[A]$ containing a and the cardinality of a (assuming that numerals are among sets or atoms).

□

Proposition 1. *For any finite transitive set* $X \subset V_\omega[A]$, *there exists a theory* T *such that* $X = Sets.\ (= Sets_T)$.

Proof. Obvious. Take

$$T = \{\ \forall x[\bigvee_{w_i \in X} x = w_i]\}.$$

□

4. Counting, cardinalities and well-orderings

In the previous section, we discussed the structure of sets and classes. Now we want to say something about counting and ordering.

4.1 Counting

Siegler and Richards [1982] describe the research results of Gelman and Gallistel on psychology of counting. Counting turns out to be a composite activity based on five principles:

1. one-one -- each object must be assigned one numeral;
2. stable order -- numerals must appear in a in a fixed order;
3. cardinal -- the last numeral corresponds to the cardinal number of the set;
4. abstraction -- anything can be counted: concrete and abstract sets, heterogeneous and homogeneous sets;
5. order irrelevance -- the pairing between objects and and numerals is arbitrary.

The five principles justify the correspondence between counting and the functional assignment of numerals to elements of a set. But we can also observe that people count using other numerals like "many", not only natural numbers. Neither is the concept of a finite set clear. Dedekind thought that a set X is finite if there is no one-to-one function from X into a proper subset of X (such an X will be called Dedekind-finite); certainly finite sets are like that, while natural numbers or reals are not. But it has been proven by Halpern and Levy (cf. [Felgner, 1971]) that (if ZF is consistent) there exist infinite Dedekind-finite sets. This shows that our counting intuitions are imperfect, and that at least in principle different counting methods could be used.
To allow for these possibilities we may simply assume that numerals form a separate entity **Nums** which is linked with sets by existence of a counting routine (function) denoted by #. We will show in Section 5 that # does have a very natural interpretation. We will allow then counting **1, 2, 3, many** , or **1, 2, 3, about_5, about_10** , where **about_5** can be defined as "any number between 4 and 6"; see Fig. 1.

Note: Cardinality function is not a set, and so $\{\#z : z \in x\}$ is not a set, unless **Nums** are admitted as sets, although $\{z : \#z = 2 \,\&\, z \in x\}$ is a well-defined set.

4.2 Orderings

We used [x , y , z] to denote the well-ordering consisting of elements x , y , z . We could have interpreted [...] also as a special function symbol in order to separate cardinalities from well-orderings; this is however not necessary. We identify [x , y , z] with { x , { y , { z } } } . Figure 2 contains a representation of a four-element well-ordering.

Definition. w ***well orders*** x , or w is a well-ordering with the *base* x , is defined as
(i) $wo(0,w) \Leftrightarrow w = 0$; $wo(\{x\},w) \Leftrightarrow w = \{x\}$
i.e. the empty set and singletons are well-ordered.

(ii) $wo(x,w) \Leftrightarrow (\exists z, y)[\, w = \{z, y\} \,\&\, z \neq y \,\&\, z \in x \,\&\, wo(\cup(x - \{z\}), y)\,]$

(y well orders the rest of x).

(iii) w is a ***well-ordering***

$$wo(w) \Leftrightarrow (\exists x)\, wo(x, w)$$

Illustration 4.2:

- Natural numbers, as defined in Section 2, are not well-orderings according to this definition.
- But if we define

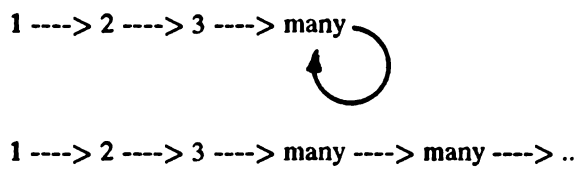


Figure 1. Numerals.
Two representations of a non-standard set of numerals **1, 2, 3, many** .

$0 = o$, $1 = \{o\}$, ... , $n + 1 = \{o, \{n\}\}$,

the new numbers are well-orderings; we have $wo(n,n)$.

We leave it as an exercise to the reader to change the above definitions to make the standard natural numbers well-orderings. For us, it will be slightly more convenient to use the non-standard definition of well-ordering. That this makes little difference can be seen from the next proposition.

□

Proposition 2. *$wo(w)$ for finite w iff w is isomorphic with a finite ordinal (i.e. natural) number.*

Proof. By induction. (Note that the isomorphism is in $V_\omega[A]$, not necessarily in *Sets*; "natural" means "as defined in Section 2").

5. The ϵ-structures

We summarize our investigation of simple common-sense set theories by proposing models for them. We then return to the two examples to show that the cardinalities and well-orderings can indeed be unrelated.

5.1 The models

We may now propose the **ϵ-structure** $\mathcal{M}(T) =$

$< V_\omega[A]$, *Sets*, ϵ, #, [ ], *Nums*, *Classes* $>$

as a model of a common-sense, unambiguous Σ_1 theory T. We may suppose that the set A of atoms is given once and for all, and that the interpretation of # and [ ] is fixed (more about it below); *Classes* and *Sets* depend on the theory T , and are given by the equations in Section 2.4.

5.2 An interpretation of cardinalities and well-orderings

The sets of standard set theories can be identified with trees whose leaves are atoms. When the axiom of regularity (foundation) is absent, the sets are directed graphs, cf. [Barwise and Etchemendy, 1987]. For the sake of simplicity, we will discuss well founded sets only. For a set s, the tree of s , $T(s)$, can be defined as $(a,b) \epsilon T(s) \iff a \epsilon b$ for a, $b \epsilon TC(\{s\})$. The edges of the tree consist here of ordered pairs of elements of the transitive closure $TC(\{s\})$. But we can imagine another, more interesting interpretation, according to which all isomorphic sets can have the same -- not just isomorphic -- edges. To this end we need a separate set of edges SE (which can be thought of, for instance, as the set of all intervals on the plane with rational endpoints).

A graph is then defined as a triple $< V, SE, E >$, where V is a set of vertices, SE is the set of edges ,

[a, b, c, d]

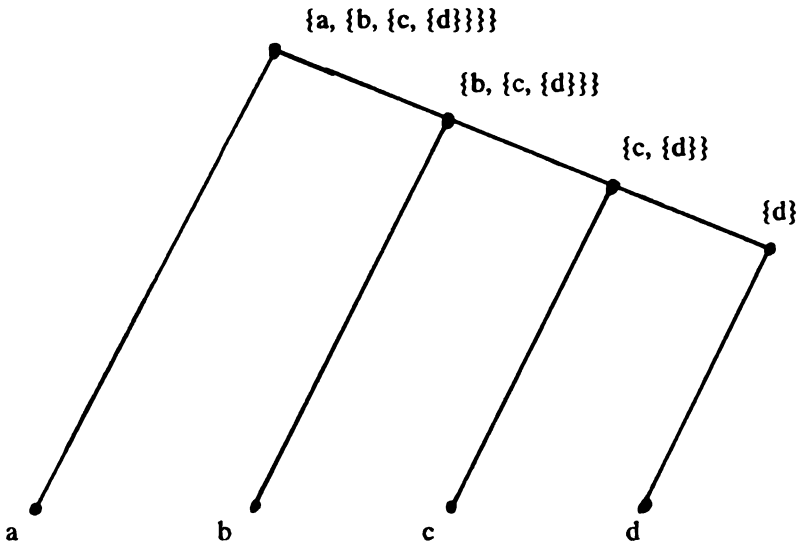


Figure 2. The graph of the well-ordering [a, b, c, d].
Well orderings are represented using nested braces. The edges represent the membership. The figure should create an impression of direction (asymmetry + linear order). Lakoff [1987, p. 362], quoting Mac Lane [1981, 198?], argues that "basic ideas from mathematics (...) correspond to the kinesthetic image schemas that arise in the study of linguistic semantics."

and E is a function from a subset of SE into $V \times V$; when (a,b) is in the range of E there is an edge between a and b. In the case of graphs of sets, we assume that $a \epsilon b$ if and only if there is an edge between a and b; see Fig.2. The edges corresponding to the **members** of the set s can be defined as

$$EM(s) = \{ e \epsilon ES : \exists v[E(e) = (v,s)] \}.$$

In classical set theory, a cardinality of a finite set s is a one-to-one function from a natural number n onto a set, i.e. it is a set of ordered pairs

$$\{ (0,x_0),(1,x_1), ...,(n-1,x_{n-1}) \},$$

where x_i's are all the elements of s. Thus all, say, three-element sets, although they have the same cardinality (namely 3), have different cardinality functions. Thus cardinality is a function from a number onto the **nodes** (or from the **nodes** onto a number).

In contrast, we define the cardinality function # as a one-to-one order preserving mapping from the **edges** $EM(s)$ of a set s into the numerals **Nums**, whose range is an initial segment of **Nums**. The last element of the range of the function is the cardinality of a given set. Under this definition if graphs of two different three-element sets are built from the same edges, the sets have identical cardinality functions. (But these functions do not belong to the *Sets* until we make an assumption about E, SE, V being sets, and make formulas defining cardinalities part of the commonsense set theory T).

Example 5.1.

Figure 3 shows a representation of a four-element set consisting of three atoms and one two-atom set. The arrows link the four edges of $EM(s)$ with some numerals. However, this mapping is not a cardinality function because it is not order-preserving; it also violates the stable order principle of Section 4.1. The other two edges of graph of the set are irrelevant for counting.

5.3 The examples

We now show that it is possible to express the non-equivalence of cardinalities and well-orderings postulated in the examples of Section 1. The examples, after an obvious formalization, serve as consistency proofs.

Definition. ϕ is *(ϵ-) consistent* iff there is an ϵ-structure

$S = < V_\omega[A], Sets, \epsilon, \#, [\ ], Nums, Classes >$

s.t. $S \models (\phi)^{Sets}$.

($(\phi)^{Sets}$ is obtained from ϕ by relativization of all quantifiers to *Sets*, as defined in Section 3.4).

Note: The relativization to the transitive class of *Sets* does not affect many important formulas; this means that a large portion of mathematics can still be done within the class of *Sets* . The concepts which are preserved are all Σ/Σ_1 formulas (by the Σ reflection principle); in particular these are: *function, relation, domain of a function/relation, range of a function/relation, pair, transitivity, ordinal, natural number, less than, successor of a set* etc. (for details see [Barwise, 1975], pp. 14, 22, 23).

Theorem 1. *Let $n > 1$, $n \in Nums$. It is (ϵ-) consistent that there is a set x with n elements which does not have a well-ordering:*

$$\exists x (\#x = n \ \& \ \neg \exists w \ wo(x,w))$$

Proof. Let n = 100. Let x = { b1 , ... , b75 , w1 , ... , w25 } . Let the theory T consist just of this assertion. Then Sets = { x } ∪ x . And clearly, no well-ordering of x belongs to Sets. The generalization for other n's is trivial.

Theorem 2. *Let $n > 1$, $n \in Nums$. It is (ϵ-) consistent that there exists a well-ordering of type n (i.e. with n elements), the elements of which do not form a set:*

$$\exists w (wo(w) \ \& \ \neg \exists x \ wo(x,w))$$

Proof. By example. Let n = 26 , let the well-ordering be given by the dots on page 2, (assuming all dots are distinct and ordered from left to right). Let the theory T consist of just one assertion
x = [a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, x, w, y, z]
then clearly x is a well-ordering (isomorphic with the dots !), but the set
{ a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, x, w, y, z } does not belong to the *Sets*.

To see this, consider a shorter well-ordering: y = [a, b, c, d]. Then *Sets* = { y , a , { b, { c, { d } } } , b, { c, { d } }, c, { d } , d } ; { a, b, c, d} $\notin$ *Sets*, (although { a, b, c, d} is a subclass of *Sets*). As before, the example generalizes easily.

Corollary 3 (for set theorists). *There exist finite proper classes.*

6. Discussion

There is perhaps no commonsense set theory; rather, there are commonsense theories of involving sets. These theories may contradict one another, and some of them may be better than others. A couple of suggestions about using collections of theories in reasoning are contained in [Zadrozny, 1987, Zadrozny 198?]. They may also apply in thinking about sets, with some additional work needed.

We have succeeded in formally describing some commonsense intuitions about cardinalities and well-orderings, but the proposed framework is not complete. In order to make it truly computational one has to solve two open problems: how to test a theory for unambiguity (or what are classes of unambiguous theories), and how to effectively compute referents of formulas. We don't have much to say about either, except to note that [Beeri *et. al*, 1987] and [Shmueli and Naqvi, 1987] show that making set theory computational may be a delicate question.

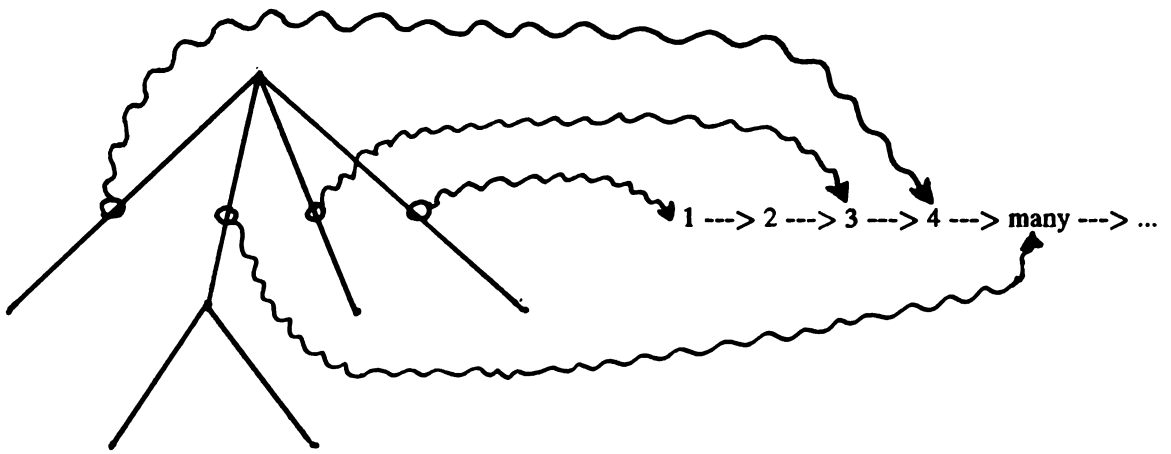


Figure 3. A function from edges of a set into numerals.
The mapping is not a cardinality function if *Nums* = {1, 2, 3, 4, *many*} . It would become one if the three arrows were shifted one unit to the left.

6.1 Reasoning with commonsense sets

Cognitive modeling/linguistics

We believe that our theory can be applied in cognitive modelling as the above examples suggest, and in computational linguistics ([Manaster-Ramer and Rounds 198?] can be an inspiration). In the latter case, one can for instance construct an automaton which would get tired after processing too long a sentence (this requires multiple **Nums**) or to represent phrases like "a group of people demonstrating at the City Hall". Then, the question "who was the first one ?" would be considered plainly nonsensical -- it would not evoke "which well-ordering do you have in mind ?". (There is no reason to assume that this set of people is well-ordered, hence there is no well-ordering of its representation, and vice versa.)

Example 6.1

The non-standard numerals **Nums** may also be useful in explaining why the sentence "I have two children" would *involve* ***fewer(n,3)*** (in the technical sense of Sadock [1975]) without involving ***fewer(n,333124)*** . Namely,

- We can suppose that a given domain D has associated with it a collection $Numalgbrs_D$ of classes of allowable numerals; for instance, D = **'human family'** would allow counting using Num_1 = **1, 2, 3, 4** , or Num_2 = **1, 2, 3, many**, or Num_3 = **1, 2, 3, about_5, about_10** ;
- The predicate *involve_n(problem,numeral)* can be defined as

$$D = Domain(problem) \;\&\; \exists\, Nums \in Numalgbrs_D \; [\, numeral \in |\, Nums\,| \,],$$

  i.e. a numeral is involved in a problem if it appears in the universe of one of the counting algebras associated with a given domain;
- In a similar fashion one can define the predicate *involve(problem,relation)* ;
- Then, since 333124 is not in any class of numerals associated with counting members of human families, while 3 appears in every such a class, the first numeral is not *involved* in building a representation for a sentence about "two children".

□

6.2 Conclusions

Common-sense reasoning about sets differs from the deductions of classical set theories. In this note, we have shown that it is possible to capture some of this reasoning in a natural, non-classical set theory. In particular, we have found a natural representation for the unordered set of balls and the uncountable set of dots from Section 1. We have also argued that it is natural to think of cardinality functions not as sets of ordered pairs, but as mappings from pointers/edges into classes of numerals. The new representations permit the separation of reasoning about orderings from reasoning about cardinalities. Similarly, we are able to explain at least some simple cases of reasoning about sets in natural language. Admittedly, the examples are simple, and for cognitive modeling the psychological connections would have to be made more precise. But it is this author's belief that, for natural language understanding, one has to partition reasoning about sets into a multitude of theories about different aspects of being a set: set algebras (dealing with union, intersection, difference), theories of cardinality (dealing with counting), theories of orderings, theories of membership etc. It is quite probable that such theories cannot be lumped together into one super-theory of commonsense reasoning about sets, but that they can be (separately) modeled in a classical set theory like ZFC or ZFC/AFA.

Acknowledgments: I'd like to thank Rohit Parikh and anonymous referees for their comments about the paper.

References

[Barwise, 1975] Jon Barwise. *Admissible Sets and Structures.* Springer, New York, 1975.

[Barwise and Etchemendy, 1987] Jon Barwise and John Etchemendy. *The Liar* . Oxford Univ. Press, New York, 1987.

[Barwise and Perry, 1983] Jon Barwise and John Perry. *Situations and Attitudes* . MIT Press, Cambridge, MA, 1983.

[Beeri *et al.*, 1987] Catriel Beeri, Shamim Naqvi, Raghu Ramakrishnan, Oded Shmueli, and Shalom Tsur. Sets and negation in a Logic Database Language (LDL1). In: *Proc. 6th ACM Conference on PODS.* San Diego, 1987.

[Breban *et al.*, 1981] M.Breban, A.Ferro, E.G. Omodeo, J.T.Schwartz. Decision Procedures for Elementary Sublanguages of Set Theory. II.Formulas Involving Restricted Quantifiers. *Communications on Pure and Applied Mathematics,* vol.34, 1981.

[Delgrande, 1987] James P. Delgrande. A Formal Approach to Learning from Examples. In: *Proc. IJCAI-87* , Milan, 1987.

[Felgner, 1971] Ulrich Felgner. *Models of ZF-Set Theory* . Springer, New York, 1971.

[Ferro *et al.*, 1980] A.Ferro, E.G. Omodeo, J.T.Schwartz. Decision Procedures for Elementary Sublanguages of Set Theory. I. Multi-Level Syllogistics and Some Extensions. *Communications on Pure and Applied Mathematics,* vol.33, 1980.

[Ferro, 1981] Alfredo Ferro. Decision Procedures for Some Classes of Unquantified Set Theoretic Formulae, Ph.D. Thesis. Dept. of Comp.Sci. New York University, 1981.

[Gregory (ed.), 1987] Richard L. Gregory (ed.).*The Oxford Companion to The Mind* .Oxford University Press, Oxford, 1987.

[Jech, 1978] Thomas J. Jech. *Set Theory.* Academic Press, New York, 1978.

[Kuper, 1987] Gabriel M. Kuper. Logic programming with sets. In: *Proc. 6th ACM Conference on PODS.* San Diego, 1987.

[Kruchten *et al.*, 1984] P.Kruchten, E.Schonberg, J.Schwartz, Software Prototyping Using the SETL Programming Language, *IEEE Software,* October 1984.

[Lakoff, 1987] George Lakoff. *Fire, Women, and Dangerous Things.* The Univ. of Chicago Press, Chicago, 1987.

[Mac Lane, 1981] Saunders Mac Lane. Mathematical Models: A Sketch for the Philosophy of Mathematics. *American Math. Monthly,* Aug.-Sept. 1981.

[Mac Lane, 198?] Saunders Mac Lane. Mathematics: Form and Function, Math. Dept., Univ. of Chicago, *to appear.*

[Manaster-Ramer and Rounds, 198?] Alexis Manaster-Ramer and William C. Rounds. On constructions, *to appear.*

[Mostowski, 1964] Andrzej Mostowski. Thirty years of foundational studies. In: K.Kuratowski *et al.*(eds.). *Andrzej Mostowski. Foundational Studies, Selected Works.* North Holland, Amsterdam, 1979.

[Ohsuga and Yamauchi, 1985] Setsuo Ohsuga and Hiroyouki Yamauchi. Multi-Layer Logic - A Predicate Logic Including Data Structure as Knowledge Representation Language. *New Generation Computing,* 3 , 1985.

[Perlis, 1987a] Donald Perlis, Commonsense set theory. In: P. Maes and D. Nardi (eds.) *Meta-Architecture and Reflection* . North-Holland, Amsterdam, 1987.

[Perlis, 1987b] Donald Perlis, Circumscribing with sets, *Artificial Intelligence* 31, 1987.

[Sadock, 1975] Jerrold M. Sadock. The Soft, Interpretive Underbelly of Generative Semantics. In: P.Cole & J.L.Morgan (eds.) *Syntax and Semantics, Vol. 3, Speech Acts* , 1975.

[Siegler and Richards, 1982] R. S. Siegler, D. D. Richards, The development of intelligence. In: R. J. Sternberg (ed.) *Handbook of Human Intelligence,* Cambridge Univ. Press, New York, 1982.

[Shmueli and Naqvi, 1987] Oded Shmueli and Shamim Naqvi. Set grouping and layering in Horn clause programs. In: *Proc. Intl. Conf. on Logic Programming,* Melbourne, 1987.

[Sternberg (ed.), 1982] Robert J. Sternberg (ed.). *Handbook of Human Intelligence,* Cambridge Univ. Press, New York, 1982.

[Vopenka, 1979] P. Vopenka. *Mathematics in the Alternative Set Theory.* Taubner, Leipzig, 1979.

[Zadrozny, 1987] Wlodek W. Zadrozny. A Theory of Default Reasoning. In: Proc. of AAAI-87. Seattle, 1987.

[Zadrozny, 198?] Wlodek Zadrozny. A Three-Level Theory of Reasoning.*submitted* .

Presentations from the Symposium on Monotonic Reasoning

Critical Issues in Nonmonotonic Reasoning

David W. Etherington
AT&T Bell Laboratories
Murray Hill, NJ

Kenneth D. Forbus
Qualitative Reasoning Group
University of Illinois

Matthew L. Ginsberg
Computer Science Department
Stanford University

David Israel
Artificial Intelligence Center
SRI International, and
Center for the Study of Language and Information

Vladimir Lifschitz
Computer Science Department
Stanford University

Abstract

The formal study of Nonmonotonic Reasoning, as we know it, has passed its tenth birthday, though its antecedents in AI go back much further. By now, the field has had a chance to develop recognizable directions and methodologies. These, in turn, have drawn adherents and critics. The panel will try to assess the state of the art, as well as suggest some alternate approaches.

The moderator begins with a short synopsis of some of the issues of concern.[1] This is followed by brief descriptions of the positions of the panelists. These are necessarily abridged—more detail and other topics will come out in the discussions.

General Issues

David W. Etherington

At times it is difficult to start reading a new paper on nonmonotonic reasoning—and yet there are so many to read. Faced with a dense formal presentation, one often wonders whether formalisms have taken on a life of their own, independent of the larger enterprises of Artificial Intelligence. Contrary to popular belief, perhaps it is *not* obvious why yet another solution to the Yale Shooting Problem is in order—not to say that it is necessarily out of order. In the following, a number of questions are offered for your approval that, while not likely to be answered by any particular paper, might profitably be kept in mind.

The first question is whether nonmonotonicity is a phenomenon or an epiphenomenon: Is it a cohesive thing to be studied in its own right, or does it more properly arise as an effect of other modalities of reasoning? Is there any reason to expect that there *are* cohesive unifying principles to be elucidated?

A related question concerns the coverage we should expect from nonmonotonic theories. Given that there is a phenomenon, should our theories be able to cover it, or will there always be examples that "fail" (whatever that means) under any particular theoretical framework? This amounts to asking whether there might be some kind of "incompleteness" result that requires any theory to fall short of treating all kinds of nonmonotonic reasoning.

Thirdly, research in the area seems to be driven largely by examples created to show up a perceived strength (or weakness) of a particular formalism (the set of such examples grows, but slowly—we still see a lot of Clyde and Tweety), rather than responding to examples/needs generated by real-life reasoning. In part, this is because existing nonmonotonic systems are too intractable to be put into practice and confronted with real-life examples, and in part because the simple examples provide a wealth of problems. But is this "inner-directedness" a liability, and is there any way to remedy it?

This brings up the issue of scaling formal nonmonotonic theories up to real problems (merely a formality?). Most extant theories are intractable—some don't even have a proof theory—and it is often difficult to tell how large bodies of information

[1] It should be kept in mind that the asking of these questions does not necessarily entail that there are no satisfactory answers!

will (or even *should*) interact. Some progress is being made in these areas, but it is ironic that formal theories addressed to the facilitation of reasoning with incomplete and seemingly-conflicting information should have such persistent problems with intractability and conflict-resolution.

Finally, how can we tell progress from motion? How can any theory of nonmonotonic reasoning be (in)validated? Are there scientific tools that can be applied to evaluate the semantical (or theoretical) basis for various frameworks, or are there only competing intuitions? Theorists in the area (notably Etherington) have been very sanctimonious about the virtue of having a semantical basis for our theories, but troubling cases like the "Clash of Intuitions" [Touretzky *et al*, 1987] give the platitudes a hollow ring. Is there really a *theory* (not just mathematics), or is there only competing intuition?

You Can't Default on Theory

Kenneth D. Forbus

My feelings about progress in commonsense reasoning are mixed. On one hand, there has clearly been substantial progress. On the other, the main source of the progress is research in qualitative physics, not the studies of default and nonmonotonic reasoning with which this panel is concerned. To me, identifying commonsense reasoning with default or nonmonotonic reasoning seems both inappropriate and inadvisable. It is inappropriate because default reasoning is ubiquitous—expert doctors, scientists, and engineers seem to use it just as often in their professional lives as when crossing the street. It is inadvisable because it misleads researchers into thinking they can find a domain-independent "silver bullet" that will solve the problem once and for all. To understand commonsense reasoning involves understanding reasoning about the physical, social, and mental worlds. While the existence of general-purpose techniques for default reasoning seems likely, they are not a substitute for having an adequate theory of the domain to be reasoned about.

I think part of the malaise in the default/nonmonotonic reasoning community is due to isolation from real examples of the very phenomena they are trying to capture. The Yale shooting non-problem is an example. The controversy seems to be that simple default reasoning techniques do not suffice to capture our intuitions about change and the world. No one should have been surprised by this. "Intuition" in this context includes what we may more neutrally call a *domain model.* Ignoring the need for such models is the root of the confusion. The gun can't unload itself? How do you know? Tomorrow's "brilliant weapons" might well opt out of killing their owner by unloading themselves. It is only our theory of guns—left out of the problem formalization—which makes this conclusion unreasonable.

It might seem surprising that studying particular domains might lead to general insights more quickly than directly attempting general formulations, but this has been the case in qualitative physics. Qualitative simulators have for years successfully reasoned about change, using closed-world assumptions and employing several different levels of knowledge to prune "anomalous extensions". Continuity, for example, is a particularly powerful constraint on systems with continuous aspects. When we get bogus results, we take it to mean that our theory, domain model, or code is inadequate, not that the whole enterprise is doomed. Last year I successfully extended my temporal inheritance techniques to include some kinds of reasoning about changes caused by actions, thus demonstrating that these techniques have promise for more general problems. There is no shortage of problems remaining, but the qualitative physics community is optimistic, unlike the sample of default reasoners I've seen lately.

I believe research on default reasoning is very important. However, if one is serious about formalizing commonsense reasoning, one must necessarily focus on that which commonsense is about. For instance, I expect progress on reasoning about general actions to rely on a better understanding of the notions of agency and causality, not on yet hairier forms of circumscription. Research that formalizes the commonsense intuitions in particular areas is a necessary complement to studies of general formulations. Such interactions can be useful to both sides: For instance, Winslett's possible models approach to reasoning about change looks like a promising way to further generalize the techniques used in qualitative physics. If the members of default reasoning community can overcome a distaste for the particular, I expect that the current crisis of confidence will subside as more progress is made.

Thoughts on Nonmonotonic Reasoning

Matthew L. Ginsberg

Let me say first that I remain as enthusiastic as ever regarding the eventual role in AI of commonsense reasoning in general and nonmonotonic reasoning in particular – both of these areas seem to have become accepted as central to the eventual success of our discipline.

I have therefore chosen to focus on what strike me as serious problems with current nonmonotonic work; briefly put, I believe that we have put ourselves in a position where it is almost impossible for our work to be validated by anyone other than a member of our small and ingrown subcommunity of AI.

The reason that this is so important is that validation is what distinguishes science from alchemy. The fundamental goal of constructing an intelligent artifact (or turning lead into gold) is a grandiose and difficult

engineering task. Enthusiastic pursuit of this ambition is fine, but if we wish to lay claim to a *scientific* pursuit, we must identify and solve subproblems that are recognizably on the critical path between us and our goal.

The early work on the formalization of commonsense reasoning was motivated by concerns such as these. More recently, however, we have lost track of our fundamental objectives. Instead of focussing on the result of applying our intuitions to a broad range of problems, we have focussed on a few trivially simple examples. Instead of remembering that we are an engineering discipline and concentrating on implementations, we have ignored our roots and concentrated solely on mathematics. (Consider the current fascination with model-theoretic approaches to nonmonotonic inference.)

I am especially troubled by what seems to me to be a trend not to formalize our intuitions, but to get the "right" answers to a small set of simple problems. What reason is there to believe that a formal system designed in this fashion will be of use to the AI community as a whole? I am aware, of course, that authors generally believe that they *have* formalized their intuitions – this is why it always strikes me as surprising when there is a wealth of counterexamples to their suggestions. How many people working on describing action have actually attempted to implement a planner using their ideas?

The issue here is one of validation. To overcome these difficulties, I would suggest that simulated domains be constructed, and simulated agents should attempt to operate in those domains.[2] These simulated agents may fare well, and they may fare poorly; either way, we will have learned something more significant than their ability to cope with a now-tired suite of simplistic examples. An effective alternative would be for us to tackle reasoning problems of interest to the industrial community.

To my mind, the nonmonotonic community needs this ability to test its hypotheses in a rich environment far more than it needs more progress on theoretical issues. But for those who insist on continuing to work in the paradigm we have developed, I would suggest the following topics:

First, we need to understand causality. Philosophers have been wrestling with this for some time (generally without success), but there is some reason to believe that we are more likely than they to make progress – we have identified some specific areas in which tractable problems appear to depend on causal notions for their solution. The analysis of inheritance hierarchies is one such area; the formal methods that have been presented here are struggling to describe conditions under which the success of one default causes the defeat of another. Reasoning about action (especially coming to grips with the qualification problem) is another area where a causal understanding of the domain is crucial.

Secondly, we need to effectively justify the eventual role of nonmonotonic reasoning within AI itself. Nonmonotonic methods were initially proposed as a way to *speed* inference by allowing a reasoning system to jump to default conclusions, but all of our formal work is computationally intractable. We need to find and understand situations in which some form of nonmonotonicity *does* lead to a computational speedup. Can default rules, for example, somehow be used to focus the search of a conventional theorem prover? Here is an application where one can imagine that a failed consistency check might not matter as much as it does in some other domains; if nonmonotonic representation techniques could be used to give a declarative formalization of heuristic control rules, that would be significant progress.

Very Brief Reflections

David Israel

Life

What are most people working on nonmonotonic systems really working on? Defeasible reasoning. What is defeasible reasoning? Here's a purely negative, partial characterization: Defeasible reasoning is reasoning in accordance with principles or rules of inference, not all of which are guaranteed to be sound with respect to truth or validity. So it's what all of us do most of the time. Some of the time, some of us do some of it well—that is, produce episodes of good defeasible reasoning. We'd like our systems to do as well. What do we mean by this?

Here's another way to ask that question: If defeasible rules or principles need not preserve truth or validity (or even consistency?), what should they preserve? How about the property of being well-justified or reasonable? A first, rough, account might go like this: Imagine that all defeasible rules take the following form:

From Γ, infer α *unless* δ.

We model beliefs by sentences in some fixed $\mathcal{L}$; α, $\delta \in \mathcal{L}$, $\Gamma \subseteq \mathcal{L}$. A defeasible rule is *good* just in case, whenever every belief in Γ is well-justified, then belief in α is, too, provided that belief in δ isn't. For any given pair $< \Gamma, \alpha >$, there could many such *defeaters*, δ. Also we are not distinguishing those defeaters that count against inferring α by supporting $\neg\alpha$ from those that count against Γ's supporting α. (See [Pollock, 1987] for more on this.)

[2] *Real* domains would obviously be better, but operation in actual domains is beyond the practical abilities of existing robotic agents. Note also that we are assuming that the designers of the agents and of the environment are different; this will *hopefully* prevent the agents from succeeding by exploiting loopholes in the simulation.

But this is all rather bloodless. Imagine our agent is interested in the question of α, has well-justified beliefs that ϕ for every ϕ in Γ, and is quite unaware of any defeaters. Then it's OK for the agent to infer α? Even if there are, in fact, lots of defeaters that it really should know about? Shouldn't it expend some energy in figuring out whether there are any defeaters? Imagine now the situation changed in the following way: the agent *as yet* has no good reason for believing any δ, though it is aware that there are counterpossibilities. How much energy should it expend on checking out those δ's of which it is aware; surely they're not likely to be equally damaging, if true. How much should it expend in figuring out whether there any others? Here the distinction between those that support $\neg\alpha$ and those that more directly undermine the justificatory connection between Γ and α might be important.

All this simply points in the direction of a crucial dimension of evaluation for a defeasible reasoning system: the cost and benefits to an agent, with given tasks to perform, in a given set of environments, of *acting* on the basis of reasonable, though perhaps false, conclusions—sometimes in the face of unexplored, though recognized, counterpossibilities. But, of course, that's *LIFE*—not *LOGIC*.

Logic

Just a brief dissent from what looks to be a bandwagon. Defeasible reasoning, even as roughly characterized, sounds like a qualitative form of probabilistic or statistical inference, together with a rule of acceptance. That sounds like an apt description of Reiter [Reiter, 1980]. But does it sound like a plausible rendering of the minimization-based systems? Does it sound like what's going in circumscription, or in various forms of the closed-world assumption? And conversely, what does defeasible reasoning, so characterized, have to do with minimizing the extensions of predicates or formulae?

Think of McCarthy's original motivating examples: the missionaries and cannibals *puzzle*; the role of induction principles and explicit second-order definitions, e.g., in characterizing the structure of the natural numbers. The last is especially critical: does the use of inductive definitions and their strengthenings—to guarantee, say, that what is being captured is the *least* fixed point—have anything to do with reasoning by statistical or probabilistic inference?

We know there are some connections: the technique of introducing, and then minimizing, *abnormality* predicates allows one partially to mimic default reasoning via minimization. And there are partial simulations in the other directions as well.

Still, we shouldn't be content with what still seems a superficial unification. Various technical results on the strength of minimization point to significant differences. So, too, does the failure of even normal default inference to be cumulatively transitive, and hence to be representable by preferential model structures. Then there is the arguably special case of default-inheritance networks—see [Thomason and Horty, 1989] for arguments to the effect that none of the standard accounts is quite right for them.

Finally, there is the phenomenon of *belief revision*, understood as encompassing cases in which some, at least, of an agent's initial beliefs or "axioms" are themselves vulnerable to revision. When modeled formally, such processes are worse than nonmonotonic; they don't even satisfy *Reflexivity*. (See [Gärdenfors, 1988] for more on belief revision.) In any case, this is surely an important phenomenon, yet none of the mainstream work on nonmonotonic reasoning touches it directly. Maybe that's because it's too close to real *LIFE*.

In conclusion:

- I second Matt Ginsberg's call for attempts at validating particular inference systems and policies in (at least) somewhat realistic situations.
- Let at least a few flowers bloom and don't be too hasty about stamping out what seem to be weeds.

In Defense of Tweety

Vladimir Lifschitz

Some members of the nonmonotonic community have expressed dissatisfaction with the slow rate of progress in this particular area of AI. It's high time, they say, for Tweety to take off, for Fred to die, and for us to move from these simplistic examples to something more impressive and maybe even useful.

It seems to me that we still have a lot to learn from Tweety and Fred. When the first nonmonotonic formalisms were proposed in the late seventies, their authors illustrated their ideas by several simple problems that would, they hoped, have elegant nonmonotonic formalizations. In many cases, such formalizations indeed have been found, but some of those early examples still haven't been formalized.

For instance, John McCarthy's 1980 circumscription paper begins with the analysis of the "missionaries and cannibals" puzzle. This example hasn't been done yet.

At the end of the same paper McCarthy remarks: "Circumscription may also be convenient for asserting that when a block is moved, everything that cannot be proved to move stays where it was. In the simple blocks world, the effect of this can easily be achieved by an axiom that states that all blocks except the one that is moved stay put. However, if there are various sentences that say (for example) that one block is attached to another, circumscription may express the heuristic situation better than an axiom." This is not merely the frame problem. This is a far more serious challenge—the "ramification problem,"

that hasn't been even tackled in the literature. (Several papers on this topic have been written, but nothing is published at this time, as far as I know.)

Maybe this is sad, but there is no point in ignoring this fact and abandoning small unsolved problems in favor of large unsolved problems.

I'd like to argue, on the other hand, that formal nonmonotonic reasoning has made remarkable progress, and that we have many reasons to be optimistic.

1. We have accumulated a precious collection of toy examples that are "expandable," in the sense that, besides the basic formulation, each of them has many enhanced versions. For instance, the Hanks/McDermott shooting story originally illustrated the temporal projection problem, but now its variants are used for exploring ideas about temporal explanation, causal anomalies, ramifications, concurrent actions, etc. This creates a fine environment for debugging and sharpening our formalization tools (although an outside observer, hearing of Fred and Tweety over and over again at each AI conference, may get the wrong impression that no progress is being made).

2. The close relation between nonmonotonic formalisms and logic programming has been demonstrated both by the theoretical analysis of the semantics of logic programs and by experiments with the nonmonotonic reasoning systems based on the ideas of logic programming. Nonmonotonic formalisms were designed as the *extensions* of first-order logic that make it more *expressive;* logic programming languages were originally meant to be the *subsets* of first-order logic that make it *tractable.* Paradoxically, systems of both kinds happen to have much in common. This is grounds for optimism regarding computational nonmonotonic reasoning. On the other hand, recent progress in understanding the declarative meaning of "negation by failure" in logic programs and deductive databases would have been impossible without the theory of nonmonotonic reasoning.

3. The best of what has been done in this area of AI strikes me as the perfect blend of pure and applied research, of mathematics and real life. The patterns of commonsense reasoning that we are trying to formalize can be easily explained without using a single technical term. But some of the work on the properties of nonmonotonic formalisms and on their relation to each other has led to nontrivial technical mathematical results. You get the same feeling as when you first learn about differential equations, or linear programming, or the analysis of algorithms—the feeling of good, clean, healthy science.

4. The mathematics of nonmonotonic reasoning is now understood much better than before, and this will have a profound effect on the methodology of research on the nonmonotonic aspects of commonsense knowledge. A few years ago, it was considered normal that the author of a paper would write a few nice-looking axioms and defaults, and stop there—without even trying to check that the formalization actually produced the expected results. He simply didn't know how to do it. Now we are beginning to see formalizations accompanied by the *theorems* that explain precisely in what sense the formalizations are sound. This approach, standard in research on the foundations of mathematics, guarantees that bugs of the simplest kind are uncovered before the paper is submitted for publication.

References

[Gärdenfors, 1988] Peter Gärdenfors. Knowledge in Flux. MIT Press, Bradford Books, Cambridge, MA, 1988.

[Pollock, 1987] John L. Pollock. Defeasible Reasoning. Cognitive Science, 11(4):481-518, October-December, 1897.

[Reiter, 1980] Raymond Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, April, 1980.

[Thomason and Horty, 1989] Richmond H. Thomason and John F. Horty. Logics for Inheritance Theory. In *Non-Monotonic Reasoning*, Reinfrank *et al.*, (eds.), pp 220–237. Springer-Verlag, Berlin, 1989.

[Touretzky *et al*, 1987] Touretzky, D., Horty, J., and Thomason, R., "A Clash of Intuitions: The Current State of Nonmonotonic Multiple Inheritance Systems", *Proc. IJCAI-87*, Milan, Italy, pp 476–482, 1987.

Probabilistic Semantics for Nonmonotonic Reasoning: A Survey

Judea Pearl
Computer Science Department
Cognitive Systems Laboratory
University of California Los Angeles, California, 90024

Abstract

The paper surveys several investigations into the possibility of establishing sound probabilistic semantics for various aspects of nonmonotonic reasoning. One such semantics, based on infinitessimal probabilities and Adams' conditional logic, is discussed at length and shown capable of serving as a universal core for a variety of dialectic-based nonmonotonic logics.

1. Why Probabilistic Semantics? Or, Conventions vs. Norms

In nonmonotonic logics, defeasible sentences are usually interpreted as conversational conventions, as opposed to descriptions of empirical reality [McCarthy 1986, Reiter 1987]. For example, the sentence "Birds fly" is taken to express a communication agreement such as: "You and I agree that whenever I want you to conclude that some bird does not fly, I will say so explicitly; otherwise you can presume it does fly." Here the purpose of the agreement is not to convey information about the world but merely to guarantee that in subsequent conversations, all conclusions drawn by the informed match those intended by the informer. Once the agreement is accepted by an agent, the meaning of the sentence acquires a dispositional character: "If x is a bird and I have no reasons to presume the contrary, then I am disposed to believe that x flies." Neither of these interpretations invokes any statistical information about the percentage of birds that fly nor any probabilistic information about how strongly the agent believes that a randomly chosen bird actually flies.

This work was supported in part by National Science Foundation Grant #IRI-8610155 and Naval Research Grant #N00014-87-K-2029.

However, the probabilistic statement $P[(Fly(x) \mid Bird(x)] = High$ (to read: "If x is a bird, then x probably flies") offers such a clear interpretation of "Birds fly", that it is hard to refrain from viewing defeasible sentences as fragments of probabilistic information. With such declarative statements it is easier to define how the fragments of knowledge should be put together coherently, to characterize the set of conclusions that one wishes a body of knowledge to entail, and to identify the assumptions that give rise to undesirable conclusions, if any.

The reasons are several. First, semantics has traditionally been defined as a relation between the speaking agent and entities external to the agent. Probabilistic information is, by its very nature, a declarative summarization of constraints in a world external to the speaker. As such, it is empirically testable (at least in principle), it is often shared by many agents, and conclusions are less subject to dispute. Second, in many cases, it is the transference of probabilistic knowledge that is the ultimate aim of common conversations, not the speaker's pattern of dispositions (which are often arbitrary). In such cases, the empirical facts that caused the agent to commit to a given pattern of dispositions are more important than the dispositions themselves, because it is those empirical facts that the listening agent is about to confront in the future. Finally, being a centuries-old science, the study of probabilistic inference has accumulated a wealth of theoretical results that provide shortcuts between the semantics and the intended conclusions. This facilitates quick generation of meaningful examples and counterexamples, quick proofs of necessity and/or impossibility, and thus, effective communication among researchers.

But even taking the extreme position that the only purpose of default statements is to establish conversational conventions, probabilists nevertheless believe that, as long as we are in the process formulating those conventions, we cannot totally ignore their empirical origin. Do-

ing so would resemble the hopeless task of formulating qualitative physics in total ignorance of the quantitative laws of physics, or, to use a different metaphor, designing speech recognition systems oblivious to the laws of phonetics.

The search for probabilistic semantics is motivated by the assumption that the conventions of discourse are not totally arbitrary, but rather, respect certain universal norms of coherence, norms that reflect the empirical origin of these conventions. Probabilistic semantics, by summarizing the reality that compelled the choice of certain conventions over others, should be capable of revealing these norms. Such norms should tell us, for example, when one convention is incompatible with another, or when one convention should be a natural consequence of another; examples of both will be illustrated in Section 4.

The benefits of adopting probabilistic norms apply not only to syntactical approaches to nonmonotonic reasoning, but also to semantical approaches, such as those based on preferential models [Shoham 1987]. Inferences based on preferential models are much less disciplined than those based on probability, because the preferences induced on possible worlds are not constrained a priori, and can, in general, be totally whimsical. Indeed, such a wide range of syntactical approaches to nonmonotonic reasoning can be formulated as variants of preferential models [Shoham 1987], that highly sophisticated restrictions must be devised to bring preferential models in line with basic standards of rationality [Lehman and Megidor 1988] (see Section 6.2).

2. Nonmonotonic Reasoning Viewed as Qualitative Probabilistic Reasoning.

To those trained in traditional logics, symbolic reasoning is the standard, and nonmonotonicity a novelty . To students of probability, on the other hand, it is symbolic reasoning that is novel, not nonmonotonicity. Dealing with new facts that cause probabilities to change abruptly from very high values to very low values is a commonplace phenomenon in almost every probability exercise and, naturally, has never attracted special attention among probabilists. The new challenge for probabilists is to find ways of abstracting out the numerical character of high and low probabilities, and cast them in linguistic terms that reflect the natural process of accepting and retracting beliefs. Thus, while nonmonotonic reasoning is commonly viewed as an extension to standard logic, it can also be viewed as an exercise in qualitative probability, much like physicists view current AI research in qualitative physics.

In research on qualitative reasoning, it is customary to discretize and abstract real quantities around a few "landmark" values [Kuipers 1986]. For example, the value 0 defines the abstraction: positive, negative and zero. In probability, the obvious landmarks are {0, ½, 1}, where 0 and 1 represent *FALSE* and *TRUE*, respectively, and ½ represents the neutral state of total ignorance. However, direct qualitative reasoning about {0, 1} reduces to propositional logic, while reasoning with the intervals [0, ½] and [½, 1] is extremely difficult — to properly process pieces of evidence and determine if a given probability should fall above ½ requires almost the full power of numerical probability calculus [Bacchus 1988].

Following the tradition of qualitative reasoning in physics and mathematics, two avenues are still available for qualitative analysis:

1. "Perturbation" analysis, to determine the direction of CHANGE induced in the probability of one proposition as a result of learning the truth of another, and

2. An "order-of-magnitude" analysis of proximities to the landmark values.

The first approach has been pursued by Wellman [1987] and Neufeld & Poole [1988], and the second by Adams [1975], Spohn [1988], Pearl [1988] and Geffner [1988].

2.1 Perturbation Analyses

Both Wellman [1987] and Neufeld & Poole [1988] investigated the logic behind the qualitative relation of *influence* or *support*, namely, the condition under which the truth of one proposition would yield an increase in the probability of another. Wellman's analysis focuses on variables with ordered domains (e.g., "An increase in quantity a is likely to cause an increase in quantity b.") as a means of providing qualitative aids to decisions, planning and diagnosis. Neufeld and Poole, focused on the relation of *confirmation* between propositions (e.g., *Quaker(Nixon)* adds confirmation to *Pacifist(Nixon)*), and viewed this relation as an important component of default reasoning.

Both approaches make heavy use of conditional independence and its graphical representation in the form of Bayesian networks or influence diagrams [Pearl 1988]. The reason is that, if we define the relation "A supports B" (denoted $S(A,B)$) as

$$S(A,B) \quad \text{iff} \quad P(B|A) \geq P(B), \qquad (1)$$

then this definition in itself is too weak to yield interesting

inferences. For example, whereas we can easily show symmetry $S(A,B) \Leftrightarrow S(B,A)$ and contraposition $S(A,B) \Leftrightarrow S(\neg B,\neg A)$, we cannot conclude cumulativity (i.e. that $S(A \wedge B, C)$ follows from $S(A,B)$ and $S(A,C)$), nor transitivity (i.e., that $S(A,C)$ follows from $S(A,B)$ and $S(B,C)$). For the latter to hold, we must assume that C is conditionally independent of A, given B,

$$P(C \mid A,B) = P(C \mid B), \qquad (2)$$

namely, that knowledge of A has no influence on the probability of C, once we know B.

Conditional independence is a 3-place nonmonotonic relationship that forms a *semi-graphoid* [Pearl and Verma 1987, Pearl 1988]. Semi-graphoids are structures that share some properties of graphs (hence the name) but, in general, are difficult to encode completely, in a compact way. The assumption normally made in probabilistic reasoning (as well as in most nonmonotonic logics, though not explicitly) is that if we represent dependence relationships in the form of a directed (acyclic) graph, then any link missing from the graph indicates the absence of direct dependency between the corresponding variables. For example, if we are given two defeasible rules, $a \rightarrow b$ and $b \rightarrow c$, we presume that a does not have any direct bearing on c, but rather, that c is independent of a, given the value of b. An important result from the theory of graphoids states that there is indeed a sound and complete procedure (called d-separation) of inferring conditional independencies from such a graph. However, this requires that the graph be constructed in a disciplined, stratified way: Every variable x should draw arrows from all those perceived to have *direct influence* on x, i.e., those that must be known to render x independent of all its predecessors in some total order (e.g., temporal). In practice, this presumes that the knowledge provider has taken pains to identify all direct influences of each variable in the system.

Neufeld and Poole have assumed that if we take isolated default statements and assemble them to form a directed graph, the resulting graph would display all the dependencies that a stratified graph would. Unfortunately, this is not always the case, and may lead to unsound conclusions. For example, from the defaults $A \rightarrow B$, $C \rightarrow \neg B$, we will conclude (using the d-separation criterion) that A is independent of C (there is no directed path between A and C). In reality, however, any two classes A and C whose members differ substantially in one typical property (B vs. $\neg B$), cannot be totally independent.

Wellman has circumvented this difficulty by starting from a well structured Bayesian network, and by defining "support" in a more restrictive way. Instead of Eq. (1), Wellman's definition reads:

$$S^+(a,b,G) \text{ iff } P(B \mid A,x) \geq P(B \mid x) \qquad (3)$$

where $S^+(a,b,G)$ stands for "a positively influences b, in the context of graph G", and the inequality should hold for every valuation x of the direct predecessors of b (in G). This stronger definition of support defines, in fact, the conditions under which inferences based on graphically-derived dependencies are probabilistically sound. Compared with the system of Neufeld and Poole, soundness is acquired at the price of a more elaborate form of knowledge specification, namely, the structure of a Bayesian network.

2.2 Infinitesimal Analysis

Spohn [1989] has introduced a system of belief revision (called OCF for Ordinal Conditional Functions) which requires only integer-value addition, and yet retains the notion of conditionalization, a facility that makes probability theory context dependent, hence nonmonotonic. Although Spohn has proclaimed OCF to be "non-probabilistic," the easiest way to understand its power and limitations is to interpret OCF as an infinitesimal (i.e., non-standard) analysis of conditional probabilities.

Imagine an ordinary probability function P defined over a set W of possible worlds (or states of the world), and let the probability $P(w)$ assigned to each world w be a polynomial function of some small positive parameter ε, for example, $\alpha, \beta\varepsilon, \gamma\varepsilon^2, \ldots$, etc. Accordingly, the probabilities assigned to any subset A of W, as well as all conditional probabilities $P(A \mid B)$, will be rational functions of ε. Now define the OCF function $\kappa(A \mid B)$ as

$$\kappa(A \mid B) = \text{lowest } n \text{ such that } \lim_{\varepsilon \to 0} P(A \mid B)/\varepsilon^n \text{ is finite}.$$

In other words, $\kappa(A \mid B) = n$ iff $P(A \mid B)$ is of the same order as ε^n, or equivalently, $\kappa(A \mid B)$ is of the same order-of-magnitude as $[P(A \mid B)]^{-1}$.

If we think of n for which $P(w) = \varepsilon^n$ as measuring the degree to which the world w is disbelieved (or the degree of surprise were we to observe w), then $\kappa(A \mid B)$ can be thought of as the degree of disbelief (or surprise) in A, given that B is true. It is easy to verify that κ satisfies the following properties:

1. $\kappa(A) = \min \{\kappa(w) \mid w \in A\}$

2. $\kappa(A) = 0$ or $\kappa(\neg A) = 0$, or both

3. $\kappa(A \cup B) = \min\{\kappa(A), \kappa(B)\}$

4. $\kappa(A \cap B) = \kappa(A \mid B) + \kappa(B)$

These reflect the usual properties of probabilistic combinations (on a logarithmic scale) with *min* replacing addition, and addition replacing multiplication. The result is a probabilistically sound calculus, employing integer addition, for manipulating order-of-magnitudes of disbeliefs. For example, if we make the following correspondence between linguistic quantifiers and ε^n :

$P(A) = \varepsilon^0$	A is believable	$\kappa(A) = 0$
$P(A) = \varepsilon^1$	A is unlikely	$\kappa(A) = 1$
$P(A) = \varepsilon^2$	A is very unlikely	$\kappa(A) = 2$
$P(A) = \varepsilon^3$	A is extremely unlikely	$\kappa(A) = 3$
----	----------	----

then Spohn's system can be regarded as a nonmonotonic logic to reason about likelihood (contrast with the modal logic of Halpern and Rabin [1987]). It takes sentences in the form of quantified conditional sentences, e.g., "Birds are likely to fly", "Penguins are most likely birds", "Penguins are extremely unlikely to fly," and returns quantified conclusions in the form of "If Tim is a penguin-bird then he is extremely unlikely to fly"

The weakness of Spohn's system, shared by numerical probability, is that it requires the specification of a complete probabilistic model before reasoning can commence. In other words, we must specify the κ associated with every world w. In practice, of course, such specification need not be enumerative, but can use the decomposition facilities provided by Bayesian networks. However, this too might require knowledge that is not readily available in common discourse. For example, using the language of defaults, we must specify $\kappa(p \mid x_1, x_2, ..., x_m)$ for each proposition p, where $x_1, x_2, ..., x_m$ represents any valuation of the antecedents of all defaults of the form $x_i \rightarrow p$. No symbolic machinery is provided for drawing conclusions from partially specified models, for example, from those that associate a κ merely with each individual default. Such machinery is provided by the conditional logic of Adams [1975], to be discussed next.

Adams' logic can be regarded as a variant of Spohn's OCF system, with input sentences specifying κ values of only 0 and 1. However, instead of insisting on a complete specification, the logic admits fragmentary statements of conditional probabilities, treats them as constraints over the distribution of κ, and infers only such statements that are compelled to acquire high likelihood by virtue of these constraints.

Due to its importance as a bridge between probabilistic and logical approaches, we will provide a more complete introduction to Adams' logic, using excerpts from Chapter 10 of [Pearl 1988]. We will see that the semantics of infinitesimal probabilities (called ε-semantics in [Pearl 1988]) leads to a two-level architecture for nonmonotonic reasoning :

1. A conservative, consistency-preserving ***core***, employing a semi-monotonic logic, and producing only inferences that are actually entailed by the input information.

2. An adventurous ***shell***, sanctioning a larger body of less grounded inferences. These inferences reflect probabilistic information that is not included in the input, yet, based on familiar patterns of discourse, can reasonably be assumed to be implicit in the input.

3. The Conservative Core

3.1 ε -Semantics

We consider a default theory $T = \langle F, \Delta \rangle$ in the form of a database containing two types of sentences: factual sentences (F) and default statements (Δ). The factual sentences assign properties to specific individuals; for example, $p(a)$ asserts that individual a has the property p. The default statements are of the type "p's are typically q's", written $p(x) \rightarrow q(x)$ or simply $p \rightarrow q$, which is short for saying "any individual x having property p typically has property q". The properties $p, q, r \cdots$ can be compound boolean formulas of some atomic predicates $p_1, p_2, ... p_n$, with x as their only free variable. However, no ground defaults (e.g., $p(a) \rightarrow q(a)$) are allowed in F and no compound defaults (e.g., $p \rightarrow (q \rightarrow r)$) are allowed in Δ. The default statement $S' : p \rightarrow \neg q$ will be called the *denial* of $S : p \rightarrow q$.

Nondefeasible statements such as "all birds are animals" will be written $Birds(x) \wedge \neg Animal(x) \rightarrow FALSE$. This facilitates the desirable distinction between a generic rule $p(x) \Longrightarrow q(x)$ (to be encoded in Δ as $p \wedge \neg q \rightarrow FALSE$) and a factual observation $p(a) \supset q(a)$, which must enter F as $\neg p \vee q$. Indeed, the information $\{p(a), p(x) \Longrightarrow q(x)\}$ will give rise to totally different conclusions (about a) than $\{p(a), \neg p(a) \vee q(a)\}$, in conformity with common use of conditionals. (A more natural treatment of nondefeasi-

ble conditionals, retaining their rule-like character, is given in [Goldszmidt and Pearl 1989]).

Let L be the language of propositional formulas, and let a *truth-valuation* for L be a function t, that maps the sentences in L to the set {1,0}, (1 for *TRUE* and 0 for *FALSE,)* such that t respects the usual Boolean connectives. To define a probability assignment over the sentences in L, we regard each truth valuation t as a world w and define $P(w)$ such that $\sum_w P(w) = 1$. This assigns a probability measure to each sentence s of L via $P(s) = \sum_w P(w)\, w(s)$.

We now interpret Δ as a set of restrictions on P, in the form of *extreme* conditional probabilities, infinitesimally removed from either 0 or 1. For example, the sentence $Bird(x) \rightarrow Fly(x)$ is interpreted as $P(Fly(x) \mid Bird(x)) \geq 1 - \varepsilon$. ε is understood to stand for an infinitesimal quantity that can be made arbitrarily small, short of actually being zero.

The conclusions we wish to draw from a theory $T = \langle F, \Delta \rangle$ are, likewise, formulas in L that, given the input facts F and the restrictions Δ, are forced to acquire extreme high probabilities. In particular, a propositional formula r would qualify as a *plausible conclusion* of T, written $F \mid\!\sim_\Delta r$, whenever the restrictions of Δ force P to satisfy $\lim_{\varepsilon \to 0} P(r \mid F) = 1$.

It is convenient to characterize the set of conclusions sanctioned by this semantics in terms of the set of facts-conclusion pairs that are entailed by a given Δ. We call this relation ε-*entailment* [1] formally defined as follows:

Definition: Let $\mathcal{P}_{\Delta,\varepsilon}$ stand for the set of distributions licensed by Δ for any given ε, i.e.,

$$\mathcal{P}_{\Delta,\varepsilon} = \left\{ P : P(v \mid u) \geq 1 - \varepsilon \quad \text{if} \quad u \rightarrow v \in \Delta \right\} \quad (4)$$

A conditional statement $S: p \rightarrow q$ is said to be ε-*entailed* by Δ, if every distribution $P \in \mathcal{P}_{\Delta,\varepsilon}$ satisfies $P(q \mid p) = 1 - O(\varepsilon)$, (i.e., for every $\delta > 0$ there exists a $\varepsilon > 0$ such that every $P \in \mathcal{P}_{\Delta,\varepsilon}$ would satisfy $P(q \mid p) \geq 1 - \delta$).

[1] Adams (1975) named this p-entailment. However, ε-entailment better serves to distinguish this from weaker forms of probabilistic entailment, Section 4.

In essence, this definition guarantees that an ε-entailed statement S is rendered highly probable whenever all the defaults in Δ are highly probable. The connection between ε-entailment and plausible conclusions, is simply:

$$F \mid\!\sim_\Delta r \quad \textit{iff} \quad (F \rightarrow r) \text{ is } \varepsilon\text{–entailed by } \Delta$$

3.2 Axiomatic Characterization

The conditional logic developed by Adams [1975] faithfully represents this semantics by qualitative inference rules, thus facilitating the derivation of new sound sentences by direct symbolic manipulations on Δ. The essence of Adams' logic is summarized in the following theorem, restated for default theories in [Geffner and Pearl 1988].

Theorem 1: Let $T = \langle F, \Delta \rangle$ be a default theory where F is a set of ground proposition formulas and Δ is a set of default rules. r is a plausible conclusion of F in the context of Δ, written $F \mid\!\sim_\Delta r$, iff r is derivable from F using the following rules of inference:

Rule 1 (Defaults) $(p \rightarrow q) \in \Delta \Longrightarrow p \mid\!\sim_\Delta q$

Rule 2 (Logic Theorems) $p \supset q \Longrightarrow p \mid\!\sim_\Delta q$

Rule 3 (Cumulativity) $p \mid\!\sim_\Delta q, p \mid\!\sim_\Delta r \Longrightarrow (p \wedge q) \mid\!\sim_\Delta r$

Rule 4 (Contraction) $p \mid\!\sim_\Delta q, (p \wedge q) \mid\!\sim_\Delta r \Longrightarrow p \mid\!\sim_\Delta r$

Rule 5 (Disjunction) $p \mid\!\sim_\Delta r, q \mid\!\sim_\Delta r \Longrightarrow (p \vee q) \mid\!\sim_\Delta r$

Rule 1 permits us to conclude the consequent of a default when its antecedent is all that has been learned. Rule 2 states that theorems that logically follow from a set of formulas can be concluded in any theory containing those formulas. Rule 3 (called *triangularity* in [Pearl 1988] and *cautious monotony* in [Lehman 1988]) permits the attachment of any established conclusion (q) to the current set of facts (p), without affecting the status of any other derived conclusion (r). Rule 4 says that any conclusion (r) that follows from a fact set (p) augmented by a derived conclusion (q) also follows from the original fact set alone. Finally, rule 5 says that a conclusion that follows from two facts also follows from their disjunction.

Some Meta-Theorems

T-1 (Logical Closure) $p \mid\!\sim_\Delta q, p \wedge q \supset r \Longrightarrow p \mid\!\sim_\Delta r$

T-2 (Equivalent Contexts) $p \equiv q, p \mid\!\sim_\Delta r \Longrightarrow q \mid\!\sim_\Delta r$

T-3 (Exceptions) $p \wedge q \mid\!\sim_\Delta r, p \mid\!\sim_\Delta \neg r \Longrightarrow p \mid\!\sim_\Delta \neg q$

T-4 (Right Conjunction) $p \mid\!\sim_\Delta r, p \mid\!\sim_\Delta q \Longrightarrow p \mid\!\sim_\Delta q \wedge r$

Some Non-Theorems:

(Transitivity) $p \supset q\ , q \mid\!\sim_\Delta r \Longrightarrow p \mid\!\sim_\Delta r$

(Left Conjunction) $p \mid\!\sim_\Delta r\ , q \mid\!\sim_\Delta r \Longrightarrow p \wedge q \mid\!\sim_\Delta r$

(Contraposition) $p \mid\!\sim_\Delta r \Longrightarrow \neg r \mid\!\sim_\Delta \neg p$

(Rational Monotony)

$$p \mid\!\sim_\Delta r,\ \text{NOT}(p \mid\!\sim_\Delta \neg q) \Longrightarrow p \wedge q \mid\!\sim_\Delta r$$

This last property (similar to CV of conditional logic) has one of its antecedents negated, hence, it does not yield new consequences from Δ. It is, nevertheless, a desirable feature of a consequence relation, and was proposed by Makinson as a standard for nonmonotonic logics [Lehman and Megidor 1988]. Rational monotony can be established within probabilistic semantics if we interpret $p \rightarrow q$ as an OCF constraint $\kappa(q \mid p) < \kappa(\neg q \mid p)$.

The reason transitivity, positive conjunction, and contraposition are not sanctioned by the ε-semantics is clear: There are worlds in which they fail. For instance, transitivity fails in the penguin example — all penguins are birds, birds typically fly, yet penguins do not. Left conjunction fails when p and q create a new condition unshared by either p or q. For example, if you marry Ann (p) you will be happy (r), if you marry Nancy (q) you will be happy as well (r), but if you marry both ($p \wedge q$), you will be miserable ($\neg r$). Contraposition fails in situations where $\neg p$ is incompatible with $\neg r$. For example, let $p \rightarrow r$ stand for *Birds* $\rightarrow$ *Fly*. Now imagine a world in which the only nonflying objects are a few sick birds. Clearly, *Bird* $\rightarrow$ *Fly* holds, yet if we observe a nonflying object we can safely conclude that it is a bird, hence $\neg r \rightarrow p$, defying contraposition.

Theorem 2 (Δ-monotonicity): The inference system defined in Theorem 1 is monotonic relative to the addition of default rules, i.e.,

$$\text{if}\quad p \mid\!\sim_\Delta r \quad \text{and} \quad \Delta \subseteq \Delta', \text{ then } p \mid\!\sim_{\Delta'} r$$

The proof follows directly from the fact that $P_{\Delta',\varepsilon} \subseteq P_{\Delta,\varepsilon}$ because each default statement imposes a new constraint on $P_{\Delta,\varepsilon}$. Thus, the logic is nonmonotonic relative to the addition of new facts (in F) and monotonic relative to the addition of new defaults (in Δ). Full nonmonotonicity will be exhibited in Section 4, where we consider weaker forms of entailment.

3.3 Consistency and Ambiguity

An important feature of the system defined by Rules 1-5 is its ability to distinguish theories portraying inconsistencies (e.g., $<p \rightarrow q, p \rightarrow \neg q>$), from those conveying ambiguity (e.g., $<p(a) \wedge q(a), p \rightarrow r, q \rightarrow \neg r>$, and those conveying exceptions (e.g., $<p(a) \wedge q(a), p \rightarrow \neg q>$).

Definition: Δ is said to be ε-*consistent* if $\mathcal{P}_{\Delta,\varepsilon}$ is non-empty for every $\varepsilon > 0$, else, Δ *is* ε-*inconsistent*. Similarly, a set of default statements $\{S_\alpha\}$ is said to be ε-*consistent with* Δ if $\Delta \cup \{S_\alpha\}$ is ε-consistent.

Definition: A default statement S is said to be *ambiguous*, given Δ, if both S and its denial are consistent with Δ.

Theorem 3 (Adams, 1975): - If Δ is ε-consistent, then a statement $S : p \rightarrow q$ is ε-entailed by Δ iff its denial $S' : p \rightarrow \neg q$ is ε-inconsistent with Δ.

In addition to Rules 1-5 of Theorem 1, the logic also possesses a systematic procedure for testing ε-consistency (hence, ε-entailment), similar to the method of truth-table proofs in propositional calculus.

Definition: Given a truth-valuation t, a default statement $p \rightarrow q$ is said to be *verified* under t if t assigns the value 1 to both p and q. $p \rightarrow q$ is said to be *falsified* under t if p is assigned a 1 and q is assigned a 0.

Theorem 4 (Adams, 1975): Let Δ be a finite set of default statements. Δ is ε-consistent iff for every non-empty subset Δ' of Δ there exists a truth-valuation t such that no statement of Δ' is falsified by t and at least one is verified under t.

When Δ can be represented as a network of default rules, the criterion of Theorem 4 translates into a simple graphical test for consistency:

Theorem 5 (Pearl, 1987a): Let Δ be a *default network*, i.e., a set of default statements $p \rightarrow q$ where both p and q are atomic propositions (or negation thereof). Δ is consistent iff for every pair of conflicting arcs $p_1 \rightarrow q$ and $p_2 \rightarrow \neg q$

1. p_1 and p_2 are distinct, and
2. There is no cycle of positive arcs that embraces both p_1 and p_2.

Theorem 5 generalizes Touretzky's (1986) consistency criterion to networks containing cycles as well as arcs emanating from negated proposition, (e.g., $\neg p \rightarrow q$).

3.4 Illustrations

To illustrate the power of ε -semantics and, in particular, the syntactical and graphical derivations sanctioned by Theorems 1, 3 and 5, consider the celebrated "Penguin triangle" of Figure 1.

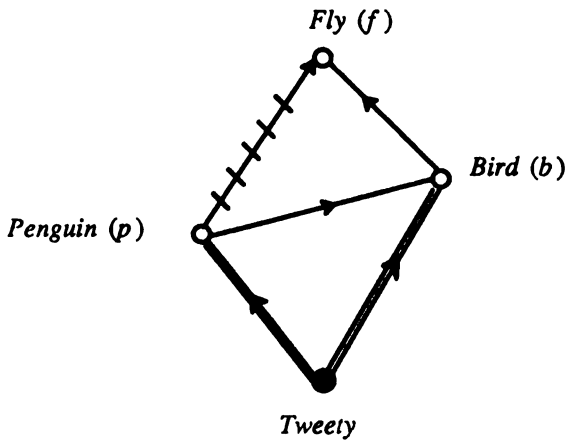


Figure 1

T comprises the sentences:

$$F = \{Penguin(Tweety), Bird(Tweety)\},$$

$$\Delta = \{Penguin \rightarrow \neg fly, Bird \rightarrow Fly, Penguin \rightarrow Bird\};$$

Although Δ does not specify explicitly whether penguin-birds fly, the desired conclusion is derived in three steps, using Rule 1 and 3 of Theorem 1:

1. $Penguin\ (Tweety) \mid\!\sim_{\Delta} \neg Fly\ (Tweety)$ (from Rule 1)
2. $Penguin\ (Tweety) \mid\!\sim_{\Delta} Bird\ (Tweety)$ (from Rule 1)
3. $Penguin\ (Tweety), Bird(Tweety) \mid\!\sim_{\Delta} \neg Fly\ (Tweety)$ (Applying Rule 3 to lines 1, 2)

Note that preference toward subclass specificity is maintained despite the defeasible nature of the rule $Penguin \rightarrow Bird$, which admits exceptional penguins in the form of non-birds.

We can also derive this result using Theorems 3 and 4 by showing that the denial of the conclusion $p \wedge b \rightarrow \neg f$ is ε-inconsistent with

$$\Delta = \{p \rightarrow \neg f, b \rightarrow f, p \rightarrow b\}.$$

Indeed, no truth-valuation of $\{p, b, f\}$ can verify any sentence in

$$\Delta' = \{p \rightarrow \neg f, p \rightarrow b, p \wedge b \rightarrow f\}$$

without falsifying at least one other sentence.

Applying theorem T-3 to the network of Figure 1 yields another plausible conclusion, $Bird \rightarrow \neg Penguin$, stating that when one talks about birds one does not have penguins in mind, i.e., penguins are exceptional kind of birds. It is a valid conclusion of Δ because every P in $\mathcal{P}_{\Delta,\varepsilon}$ must yield $P(p \mid b) = O(\varepsilon)$. Of course, if the statement $Bird \rightarrow Penguin$ is artificially added to Δ, inconsistency results; as ε diminishes below a certain level (1/3 in our case), $\mathcal{P}_{\Delta,\varepsilon}$ becomes empty. This can be predicted from purely topological considerations (Theorem 5), since adding the arc $Bird \rightarrow Penguin$ would create a cycle of positive arcs embracing "bird" and "penguin", and these sprout two conflicting arcs toward "fly". Moreover, Theorem 3 implies that if the network becomes inconsistent by the addition of S then that network ε-entails its denial, S'. Hence, the network of Figure 1 ε-entails $Bird \rightarrow \neg Penguin$. By the same graphical method one can easily show that the network also ε-entails the natural conclusion, $Fly \rightarrow \neg Penguin$. This contraposition of $Penguin \rightarrow \neg Fly$ is sanctioned only because the existence of flying objects that are not penguins (i.e., normal birds) is guaranteed by the other rules in Δ.

4. The Adventurous Shell

The preceding adaptation of Adams' logic of conditionals yields a system of defeasible inference with rather unique features:

1. The system provides a formal distinction between exceptions, ambiguities and inconsistencies and offers systematic methods of maintaining consistency in databases containing defaults.

2. Multiple extensions do not arise and preferences among arguments (e.g., toward higher specificity) are respected by natural deduction.

3. There is no need to specify abnormality relations in advance (as in circumscription); such relations (e.g., that penguin are abnormal birds) are automatically inferred from the input.

However, default reasoning requires two facilities: one which forces conclusions to be retractable in the light of new refuting evidence; the second which protects conclusions from retraction in the light of new but irrelevant evidence. Rules 1-5 excel on the first requirement but fail on the second (The opposite is true in default logics). For instance, in the example Fig. 1, if we are told that Tweety is also a blue penguin, the system would retract all previ-

ous conclusions (as ambiguous), even though there is no rule which in any way connects color to flying.

The reason for this conservative behavior lies in our insistence that any issued conclusion attains high probability in *all* probability models licensed by Δ and one such model reflects a world in which blue penguins do fly. It is clear that if we want the system to respect the communication convention that, unless stated explicitly, properties are presumed to be *irrelevant* to each other, we need to restrict the family of probability models relative to which a given conclusion must be checked for soundness. In other words, we should consider only distributions which minimize dependencies relative to Δ, i.e., they embody dependencies which are absolutely implied by Δ, and none others.

4.1 The Maximum-Entropy (ME) Approach

A traditional way of defining a minimal dependency extension to a given set of constraints is to invoke the maximum-entropy (ME) method [Jaynes 1979]. The method amounts to selecting from $\mathcal{P}_{\Delta,\varepsilon}$ a single-distribution, $P^*_{\Delta,\varepsilon}$, defined by

$$H(P^*_{\Delta,\varepsilon}) \geq H(P) \quad \forall P \in \mathcal{P}_{\Delta,\varepsilon}$$

where $H(P)$ is the entropy function

$$H[P(w)] = -\sum_w P(w) \log P(w)$$

The definition of entailment would then invoke $P^*_{\Delta,\varepsilon}$ instead of $\mathcal{P}_{\Delta,\varepsilon}$, and would yield:

Definition: A theory $T = <F, \Delta>$ *weakly-entails* a conclusion r, written $F \mid\!\sim^z_\Delta r$, iff

$$\lim_{\varepsilon \to 0} P^*_{\Delta,\varepsilon}(r \mid F) = 1 .$$

ε-entailment clearly subsumes weak entailment, because $P^*_{\Delta,\varepsilon} \in \mathcal{P}_{\Delta,\varepsilon}$.

When applied to small default system, the *ME* method yields patterns of reasoning which are rather pervasive in common discourse. For example, if a theory T involves only three primitive propositions p, q, and r, the *ME* approach gives rise to the following patterns of reasoning:

(1) Accepting Irrelevant Properties (Strengthening the Antecedents)

$$\text{If} \quad \Delta = \{q \to r\}, \quad \text{then} \quad q \wedge p \mid\!\sim^z_\Delta r$$

(2) Mediated Inheritance (Weak Transitivity)

$$\text{If} \quad \Delta = \{p \to q, q \to r\}, \quad \text{then} \quad p \mid\!\sim^z_\Delta r$$

(3) Left Conjunction

$$\text{If} \quad \Delta = \{p \to r, q \to r\}, \quad \text{then} \quad p \wedge q \mid\!\sim^z_\Delta r$$

(4) Contraposition

$$\text{If} \quad \Delta = \{p \to r\}, \text{then} \quad \neg r \mid\!\sim^z_\Delta \neg p$$

Applying the *ME* principle to larger systems reveals intriguing phenomena and challenging possibilities. For example, if the link in Figure 1 between *Penguin* and *Bird* is mediated by an intermediate property, say *Winged animal*, the conclusion $\neg Fly(Tweety)$ still follows from $F = \{Penguin(Tweety), Bird(Tweety)\}$. In other words, the intermediate property seems not to weaken the cumulativity rule which gives priority to subclasses over superclasses. Strangely, however, the conclusion $Bird(Tweety)$ no longer follows from $F = \{Penguin(Tweety)\}$. Two competing arguments (paths) now lead from *Penguin* to *Bird*; transitivity sanctions the path *Penguin* $\to$ *Winged animal* $\to$ *Bird*, and contraposition sanctions *Penguin* $\to \neg Fly \to \neg Bird$. As a result, Tweety's "birdness" becomes ambiguous.

4.2 Dialectic Approaches

The ME approach has several shortcomings, one being its improper handling of causation [Hunter 1988, Pearl 1988], the second being its computational complexity; nobody yet has been able to extract from this semantics a complete system of qualitative axioms similar to those encapsulating ε-semantics.

Dialectic approaches attempt to supplement the probabilistic interpretation of defaults with a set of assumptions about conditional independence drawn on the basis of the syntactic structure of Δ. For a default $p \to q$, these approaches assume the probability of q to be high not only when p is all that is known, but also in the presence of an additional body of evidence which does not provide an *argument* against q. This interpretation is closer in spirit to the syntactic approaches to non-monotonic reasoning proposed by Reiter [1980] and McDermott and Doyle [1980], which allow to infer q from p in the absence of 'proofs' for $\neg q$.

In the systems reported in [Geffner and Pearl 1987] and [Geffner 1988] these ideas take the form of an additional inference rule, similar to:

Rule 6: Irrelevance

If $p \rightarrow r \in \Delta$ and $I_\Delta(q, \neg r | p)$, then $p \wedge q \mid\!\sim_\Delta r$,

where the predicate $I_\Delta(q, \neg r | p)$, which reads: "q is irrelevant to $\neg r$ given p," expresses the conditional independence $P(r|p) = P(r|p, q)$.

The mechanism for evaluating the irrelevance predicate $I_\Delta(q, \neg r | p)$ appeals to the set ψ' of wffs formed by converting each default $p \rightarrow q$ in Δ into a corresponding material implication $p \supset q$. In essence, q is then said to be *relevant* to $\neg r$ given p, if there is a a set ψ' of implications in ψ which permit an *argument* for $\neg r$ to be constructed, i.e if $\psi', p, q \vdash \neg r$, with the set of wffs $\psi' \cup p, q$ being logically consistent. The set ψ' is called the *support* of the argument for $\neg r$. If q is not relevant to $\neg r$ given p, then q is assumed to be *irrelevant to* $\neg r$ in that context, and $I_\Delta(q, \neg r | p)$ thus holds.

This simple extension permits us to infer, for instance, that red birds are likely to fly given a default stating that birds fly, as 'redness' does not induce any argument in support of not flying. Further refinements are installed to insure that arguments for $\neg r$ that are blocked by p (or its consequences) do not bear on the predicate I_Δ. With this refinement, most examples analyzed in the literature yield the expected results.

Dialectic approaches constitute an alternative way of extending the inferential power of the core set of probabilistic rules. An advantage of these approaches over those based on maximum entropy is intelligibility: derivations under this approach can usually be justified in a more natural fashion. On the other hand, these approaches lack the foundational basis of a principle like maximum entropy, making it difficult to justify and make precise the form these extensions should take.

5. Do People Reason with Extreme Probabilities (or Lotteries and other Paradoxes of Abstraction)

Neufeld and Poole [1988] have raised the following objection (so-called "Dingo Paradox") in connection with the theorem of exceptions (T-3). We saw that the penguin triangle (Fig. 1) sanctions the conclusion *Bird* $\rightarrow \neg$ *Penguin* by virtue of the fact that penguins are an exceptional class of birds (relative to flying). Similarly, if "sandpipers" are birds that build nests in sand, we would conclude *Bird* $\rightarrow \neg$ *Sandpiper*. Continuing in this manner through all types of birds, and assuming that every subclass of birds has a unique, distinguishing trait, we soon end up with the conclusion that birds do not exist — birds are not penguins, not sandpipers, not canaries..., thus ruling out all types of birds.

This paradox is a variant of the celebrated Lottery Paradox [Kyburg 1961]: Knowing that a lottery is about to have one winner is incompatible with common beliefs that each individual ticket is, by default, a loser. Indeed, the criterion provided by ε-semantics would proclaim the overall set of such statements ε-inconsistent, since the set of conditional probabilities

$$P(Loser(x) | Ticket(x)) \geq 1 - \varepsilon \quad x = 1, 2, ..., N$$

cannot be satisfied simultaneously for $\varepsilon < 1/N$. Perlis [1987] has further shown that every default logic is bound to suffer from some version of the lottery paradox if we insist on maintaining deductive closure among beliefs.

Are these paradoxes detrimental to ε-semantics, or to nonmonotonic logics in general? I would like to argue that they are not. On the contrary, I view these paradoxes as healthy reminders that in all forms of reasoning we are dealing with simplified abstractions of real-world knowledge, that we may occasionally step beyond the boundaries of one abstraction and that, in such a case, a more refined abstraction should be substituted.

Predicate logic and probability theory are two such abstractions, and ε-semantics offers an abstraction that is somewhere between logic and probability. It requires less input than probability theory (e.g., we need not specify numerical probabilities), but more input than logic (e.g., we need to distinguish between defaults, $a \rightarrow b$, and facts, $\neg a \vee b$). It is more conservative than logic (e.g., it does not sanction transitivity), but more adventurous than probability theory (e.g., it admits conclusions even if their probabilities approach 1 very slowly, such as $= (1 - \varepsilon)^N$.

Each abstraction constitutes an expedient simplification of reality, tailored to serve a specialized set of tasks. Each simplification is supported by a different symbol processing machinery and by a set of norms, to verify whether the simplification and its supporting machinery are still applicable. The lottery paradox represents a situation where ε-semantics no longer offers a useful abstraction of reality. Fortunately, however, the consistency norms help us identify such situations in advance, and alert us (in case our decisions depend critically on making extensive use of the disjunction axiom) that a finer abstraction should be in order (perhaps OCF or full

ledge probability theory).

Probabilities that are infinitesimally close to 0 and 1 are very rare in the real world. Most default rules used in ordinary discourse maintain a non-vanishing percentage of exceptions, simply because the number of objects in every meaningful class is finite. Thus, a natural question to ask is, why study the properties of an abstraction that applies only to extreme probabilities? Why not develop a logic that characterizes moderately high probabilities, say probabilities higher than 0.5 or 0.9 — or more ambitiously, higher than α, where α is a parameter chosen to fit the domains of the predicates involved? Further, why not develop a logic that takes into account utility information, not merely probabilities, thus formalizing reasoning about actions, in addition to beliefs [Doyle 1988]?

The answer is that any such alternative logic would be extremely complicated; it would need to invoke many of the axioms of arithmetics, and would require more information than is usually available. Almost none of the patterns of reasoning found in common conversation will remain sound relative to such semantics. Take, for example, the logic of "majority," namely, interpreting the default rule $a \rightarrow b$ to mean "The majority of a 's are b 's," or $P(b \mid a) > 0.5$. Only the first two axioms of Theorem 1 remain sound in this interpretation. Even the cumulativity axiom, which is rarely disputed as a canon of default reasoning, is flatly violated by some proportions (e.g., $(p \wedge q)/p = 51\%$, $(p \wedge r)/p = 51\%$, $\neg q \wedge \neg r = \emptyset$, giving $(p \wedge q \wedge r)/(p \wedge q) = 2\%$.

How, then, do people reason qualitatively about properties and classes, proportions and preferences? It appears that, if the machinery invoked by people for such tasks stems from approximating numerical information by a set of expedient abstractions, then the semantics of extreme probabilities is one of the most popular among these abstractions. The axioms governing this semantics (i.e., Rules 1-5, Theorem 1) appears to have been thoroughly entrenched as inference rules in plausible reasoning. For example, from the sentences "Most students are males" and "Most students will get an A," the cumulativity axiom would infer "Most male students will get an A." This conclusion can be grossly incorrect, as shown in the example above, yet it is a rather common inference made by people, given these two inputs. Separating utilities from probabilities is another useful abstraction, commonly used in reasoning about actions.

Important information about the logic chosen by people to reason about proportions and actions is provided by many so called "paradoxes" of statistics. Take, for instance, the celebrated Simpson's Paradox [Simpson 1951]. It involves a hypothetical test of the effectiveness of a certain drug on a population consisting of males and females, and the numbers are contrived so that this drug seems to work on the population as a whole, but it has an adverse effect on males and an adverse effect on females. A person's first reaction would normally be that of surprise. Only when we look at the numbers and agree to interpret the phrase "has a positive effect" as a statement about an increase in the ratio of recovery to non-recovery cases do we begin to see that the calculus of proportions clashes with our intuitive predictions. The surprise with which people react to Simpson's Paradox suggests that the disjunction axiom (Rule 5) has been adopted as one of the canons of plausible reasoning. While this axiom is not sound relative to the semantics of increased proportions (which is also the semantics of "support" as in Eq. (1)), it is sound relative to the ε-semantics. In conclusion, it appears that the machinery of plausible reasoning is more in line with the rules of "almost-all" logic than with those of "support" or "majority" logics.

6. Relations to Possible Worlds Approaches

6.1 The Most Probable World Approach

A straightforward way of relating probabilistic methods to possible worlds approaches is to assume that we have a probability function P defined on the set W of possible worlds, and that at any state of knowledge, beliefs are fully committed to a world that has the highest probability. Formally, let $W^* \subseteq W$ be the set of most probable worlds,

$$W^* = \{w^* \mid P(w^*) \geq P(w) \; \forall \, w \in W\} \, ,$$

A proposition A is *believed* if A holds in some $w^* \in W^*$. To maintain coherence, we also demand that any set of propositions that are simultaneously believed, must hold in the same w^*. Non-monotonic behavior is obtained by conditionalization; given a body of evidence (facts) e, the probability function $P(w)$ shifts to $P(w \mid e)$ and this yields a new set of most probable worlds

$$W_e^* = \{w^* \mid P(w^* \mid e) \geq P(w \mid e) \; \forall \, w \in W\}$$

which, in turn, results in a new set of beliefs.

This approach was explored in [Pearl 1987b] where a world was defined as an assignment of values to a set of interdependent variables (e.g., assignment of *TRUE* - *FALSE* values to a set of diseases in medical diagnosis), and the worlds in W^* were called *most probable explanations* (MPE). It was shown that the task of finding a most probable explanation to a body of evidence is no more complex than that of computing the probability of a single

proposition. In singly-connected networks (directed trees with unrestricted orientation) the task can be accomplished in linear time, using a parallel and distributed message-passing process. In multiply-connected networks, the problem is NP-hard, however, clustering, conditioning and stochastic simulation techniques can offer practical solutions in reasonable time if the network is relatively sparse. Applications to circuit diagnosis are described in [Geffner & Pearl 1987, Pearl 1988].

The MPE approach provides a bridge between probabilistic reasoning and nonmonotonic logic. Like the latter, the method provides systematic rules that lead from a set of factual sentences (the evidence) to a set of conclusion sentences (the accepted beliefs) in a way that need not be truth-preserving. However, whereas the input-output sentences are categorical, the medium through which inferences are produced is numerical, and the parameters needed for complete specification of $P(w)$ may not be readily available. In modeling digital circuits this problem is not too severe, because all internal relationships are provided with the system's specifications. However, in medical diagnosis, as well as in reasoning about everyday affairs, the requirement of specifying a complete probabilistic model is too cumbersome and can be justified only in cases where critical decisions are at stake.

6.2 Relation to Preferential Model Semantics

The preferential models approach to nonmonotic reasoning [Shoham 1987] leaves room for widely different interpretations of defaults, ranging from the adventurous to the conservative. The adventurous approach takes the statement $A \rightarrow B$ to mean: Every world where $A \wedge B$ holds has a prima facie preference over the corresponding world where $A \wedge \neg B$ holds, everything else being equal (the terms "world" and "models" are used interchangeably in the literature). Conflicts are later resolved by extra logical procedures [Selman and Kautz 1988]. The conservative school [Lehman and Megidor 1988] [Delgrande 1988] takes $A \rightarrow B$ to be a faint reflection of a pre-existing preference relation, saying merely: B holds in all the most preferred worlds among those compatible with A. Whether a collection of such faint clues is sufficient to reveal information (about the preference relation) that entails a new statement $x \rightarrow y$, depends on the type of restrictions the preference relation is presumed to satisfy.

Recently, Lehman and Megidor [1988] have identified a restricted class of preferential models, whose entailment relation satisfies a reasonable set of rationality requirements. In essence, the restriction is that states of worlds be *ranked* (e.g., by some numerical score r) such that a state of higher rank is preferred to a state of lower rank. Lehman and Megidor proved that the entailment relation induced by this class of ranked models coincides exactly with Adams' ε-entailment relation defined in Eq. (4) and, of course, its properties coincide with Rules 1 through 5 of Theorem 1.

It is remarkable that two totally different interpretations of defaults yield identical sets of conclusions and identical sets of reasoning machinery. (Note that, even if we equate rank with probability, the interpretation $P(B|A) > 1 - \varepsilon$ is different from the preferential interpretation, because, for any finite ε, the former permits the most probable world of A to be incompatible with B). Based on this coincidence, it is now possible to transport shortcuts and intuitions across semantical lines. For example, Theorem 3 establishes a firm connection between preferential entailment and preferential consistency. Similarly, Theorems 4 and 5 determine the complexity of proving entailment in preferential models.

Acknowledgements

I am grateful to Ernest Adams, Hector Geffner, Daniel Lehman, David Makinson and Paul Morris for providing helpful comments on several topics of this paper.

References

[Adams 1975] Adams, E. 1975. *The logic of conditionals.* Dordrecht, Netherlands: D. Reidel.

[Bacchus 1988] Bacchus, F. 1988. Representing and reasoning with probabilistic knowledge. PhD. Dissertation, Dept. of Computer Science, University of Alberta, Edmonton.

[Delgrande 1988] Delgrande, J.P. 1988. An approach to default reasoning based on first-order conditional logic; revised report. *Artificial Intelligence*, 36:63-90.

[Doyle 1988] Doyle, J. 1988. On universal theories of defaults. Technical Report CMU-CS-88-11, Carnegie Mellon University.

[Geffner 1988] Geffner, H. 1988. On the Logic of defaults. *Proc.of the Natl. Conf. on AI (AAAI-88)*, St. Paul, 449-454.

[Geffner and Pearl 1988] Geffner, H., & Pearl, J. 1988. A framework for reasoning with defaults. UCLA Cognitive Systems Laboratory, Technical Report 870058

(R-94), March 1988. To appear in *Proc. Soc. for Exact Philosophy Conf.*, Rochester, New York, June 1988.

[Goldszmidt and Pearl 1989] Goldszmidt, M. and Pearl, J. 1989. Deciding consistency of databases containing defeasible and strict information. UCLA Cognitive Systems Laboratory, Technical Report (R-122).

[Halpern and Rabin 1987] Halpern, J. Y., and Rabin, M. 1987. A logic to reason about likelihood. *Artificial Intelligence* 32 (no. 3):379-406.

[Hunter 1989] Hunter, D. 1989. Causality and maximum entropy updating. *Intl. Journal of Approximate Reasoning*. 3 (no. 1): 87-114.

[Jaynes 1979] Jaynes, E. T. 1979. Where do we stand on maximum entropy? In *The maximum entropy formalism*, ed. R. D. Levine and M. Tribus. Cambridge: MIT Press.

[Kuipers 1986] Benjamin Kuipers. 1986. Qualitative simulation. *Artificial Intelligence*, 29:289-338.

[Kyburg 1961] Kyburg, H. E. 1961. *Probability and the logic of rational belief.* Middleton, Conn.: Wesleyan University Press.

[Lehman and Megidor 1988] Lehman, D., and Megidor, M. 1988. Rational logics and their models: a study in cumulative logics. Dept. of Computer Science, Hebrew University, Jerusalem, Israel, Technical Report #TR-88-16.

[McCarthy 1986] McCarthy, J. 1986. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence* 28 (no. 1):89-116.

[McDermott 1980] McDermott, D. V., and Doyle, J. 1980. Non-monotonic logic 1. *Artificial Intelligence* 13 (no. 1,2):41-72.

[Neufeld and Poole 1988] Neufeld, E. and Poole, D. 1988. Probabilistic semantics and defaults. *Proc. 4th AAAI Workshop on Uncertainty in AI*, Minneapolis, 275-281.

[Pearl 1987a] Pearl, J. 1987. Deciding consistency in Inheritance Networks. UCLA Cognitive Systems Laboratory, Technical Report 870053 (R-96).

[Pearl 1987b] Pearl, J. 1987. Distributed revision of composite beliefs. *Artificial Intelligence* 33 (no. 2):173-215.

[Pearl 1988] *Probabilistic reasoning in intelligent systems: networks of plausible inference.* 1988. San Mateo: Morgan Kaufmann.

[Pearl and Verma 1987] Pearl, J., and Verma, T. S. 1987. The logic of representing dependencies by directed graphs. *Proc., 6th Natl. Conf. on AI (AAAI-87)*, Seattle, 374-79.

[Perlis 1987] Perlis, D. 1987. On the consistency of commonsense reasoning. *Computational Intelligence*, 2:180-190.

[Reiter 1980] Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13:81-132.

[Reiter 1987] Reiter, R. 1987. Nonmonotonic reasoning. In *Annual review of computer science*, vol. 2, 147-86. Palo Alto, Calif.: Annual Reviews.

[Selman and Kautz 1988] Selman, B. and Kautz, H. 1988. The complexity of model-preference default theories. *Proc. CSCSI-88*, Edmonton, Alberta, 102-109.

[Shoham 1987] Shoham, Y. 1987. Nonmonotonic logics: meaning and utility. *Proc. of Intl. Joint Conf. on AI (IJCAI-87)*, Milan, 388-393.

[Simpson 1951] Simpson, E. H. 1951. The interpretation of interaction in contingency tables. *J. Royal Statist. Soc.*, ser. B 13:238-41.

[Spohn 1988] Spohn, W. 1988. A general non-probabilistic theory of inductive reasoning, *Proc. 4th Workshop on Uncertainty in AI*, Minneapolis, 315-322.

Touretzky 1986] Touretzky, D. 1986. *The mathematics of inheritance systems.* Los Altos, Calif.: Morgan Kaufmann.

[Wellman 1987] Wellman, M. 1987. Probabilistic semantics for qualitative influences. *Proc.of the Natl. Conf. on AI (AAAI-87)*, Seattle, 660-664.

Subject Index

Author Index